UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo al Egreso de Profesionales en Actividad

TÍTULO: Implementación de un Sistema para el Registro de Cuentas dentro del Gobierno Nacional

AUTOR: Guadalupe Martín

DIRECTOR ACADÉMICO: Javier Bazzocco

DIRECTOR PROFESIONAL: Belena Zulaica – Alfredo Fidani

CARRERA: Licenciatura en Sistemas

Resumen

SIRECO es un sistema implementado por la DGSIAF dentro del Ministerio de Economía. Es utilizado por los Organismos y la Tesorería General de la Nación para llevar el Registro de Cuentas Oficiales.
Es un sistema web que se vincula con e-Sidif (Sistema Integrado de Administración Financiera) para obtener información actualizada para la gestión de las cuentas oficiales.

Palabras Clave

Registro de Cuentas Oficiales, Metodologías Ágiles, Scrum, DevOps, Design Thinking, GitLab.

Conclusiones

Con la utilización de nuevas tecnologías, herramientas y los procesos de trabajo utilizados, entre otros; dejó un precedente para el resto de las aplicaciones futuras dentro de la Organización. A su vez, trajo a los usuarios un cambio para la gestión de cuentas oficiales, entre los que se destacan: cambio tecnológico, comunicación con e-Sidif, inclusión de los organismos.

Trabajos Realizados

Se describió el proyecto SIRECO, mostrando sus comienzos, junto con los distintos procesos definidos para obtener un mejor resultado, metodologías ágiles utilizadas e interacción entre los distintos equipos.

Trabajos Futuros

Como trabajo a futuro se puede ver algunos más importantes como la actualización de datos de entidades básicas de SIRECO proveniente de otros sistemas, interoperabilidad con Cuentas Operativas de e-Sidif, validación de la cuenta bancaria con Interbanking, entre otras.

Fecha de la presentación: 09/2021

UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Implementación de un Sistema para el Registro de Cuentas dentro del Gobierno Nacional

Alumna: Guadalupe Martín

Director: Javier Bazzocco

Capítulo 1

1.1 Motivación

En la Dirección General de Sistemas Informáticos de Administración Financiera (de ahora en adelante DGSIAF)[4] se desarrolla y mantiene el sistema e-Sidif y varios sistemas vinculados. Siendo este uno de los sistemas más grandes del Ministerio de Economía, el cual suplanta al Sidif Central y SIDIF Local Unificado (SLU). Hace unos años atrás, se consolidaron sistemas más pequeños, los cuales tenían y/o tienen una vinculación con e-Sidif.

Uno de estos sistemas vinculados con e-Sidif es el nuevo Sistema de Registro de Cuentas Oficiales del Sector Público Nacional (SIRECO). Los titulares de estas cuentas son llamados Organismos (Empresas, Fondos Fiduciarios, Universidades, Otros Entes y Organismos de la Administración Pública Nacional).

La aplicación que se presenta en este trabajo, nace a partir de la necesidad de reemplazar un sistema ya existente, el cual no cumplía con distintos requerimientos necesarios para el registro de cuentas. Partiendo de que la única que tenía acceso al sistema SIRECO era la Tesorería General de la Nación (Tesorería). La cual era la encargada de registrar y administrar cuentas de los distintos organismos. Por otro lado, debía ingresarlas en el sistema e-Sidif lo cual implicaba realizar una doble carga de datos.

El nuevo sistema propuesto permite agilizar el proceso de registro y administración de cuentas, ya que no sólo queda a cargo de la Tesorería sino también dejando que los titulares de las cuentas puedan realizar distintas operaciones sobre las mismas. Por otra parte, agrega algo muy importante para poder llevar un orden en la información, que es la comunicación con e-Sidif.

En este trabajo se presenta el nuevo Sistema de Registro de Cuentas Oficiales del Sector Público Nacional donde se muestra el resultado obtenido partiendo de las necesidades que los usuarios plantearon en distintas reuniones. Y así poder presentar el camino recorrido desde el inicio hasta la puesta en marcha del aplicativo nombrando:

- Tecnologías y herramientas utilizadas, donde alguna de las cuales se utilizaron por primera vez en la Organización.
- Procesos y estrategias de trabajo utilizando metodologías ágiles.
- Realizar una descripción, etapas, alcance y temas destacados dentro del sistema.
- Función de cada equipo de trabajo y su interacción con el resto de los grupos como así también la dinámica con los usuarios. En este proyecto no sólo se involucraron los equipos de análisis, UX (User Experience), UI (User Interface), desarrollo, testing, testing automático, CAU (Centro de Asistencia al Usuario), entre otros sino que también se tuvo presente al usuario, haciéndolo partícipe en cada instancia del nuevo sistema.

1.2 Objetivo

Según el artículo 78, punto 5.1 del Anexo del Decreto 1344/2007[1], Reglamentario de la Ley 24.156[2] de Administración Financiera y de los Sistemas de Control del Sector Público Nacional, la Tesorería General de la Nación (de ahora en más Tesorería), debe ser la responsable de tener una base de datos actualizada con todas las cuentas bancarias informadas y/o autorizadas, cuyos titulares son los organismos del Sector Público Nacional.

El decreto anteriormente mencionado (1344/2007) establece que *"la Tesorería General de la Nación mantendrá una base de datos denominada Registro de Cuentas Oficiales"*, de ahí nace el nombre de la aplicación, Sistema de Registro de Cuentas Oficiales (SIRECO).

Por otra parte, la Tesorería aprobó, en la Disposición N° 5/2010 TGN[3], el *"Procedimiento para autorizar la apertura, modificación y cierre de Cuentas Oficiales"*, a fin de facilitar el ingreso y actualización de los datos de la nómina de cuentas del Registro de Cuentas Oficiales.

La Tesorería ya contaba con un sistema para realizar el registro de las cuentas, pero éste tenía varias desventajas:

- La única que tenía acceso al sistema era la propia Tesorería, por lo cual a la hora de actualizar los datos se debía confeccionar una planilla Excel que contenía el stock de cuentas de cada organismo de la Administración Pública Nacional. Cada uno de

los mencionados organismos debía revisar y modificar las cuentas en el Excel y para luego devolver el archivo modificado a la Tesorería. Posteriormente la Tesorería realizaba las modificaciones indicadas por los titulares de las cuentas en el sistema.

- Redundancia de datos, ya que al no tener una vinculación con el Sistema Integrado de Administración Financiera (e-Sidif)[1], en la creación o modificación de una cuenta se necesitaban datos que se encuentraban en e-Sidif. Para este caso se había replicado diferente información, la cual quedaba desactualizada ya que si se realizaban cambios en e-Sidif no se veían reflejados en el sistema SIRECO.

El objetivo de esta tesina, se refleja en generar y ofrecer un nuevo sistema que busca mejorar la operatoria del registro de cuentas oficiales, dando la posibilidad de que cada organismo pueda chequear sus cuentas, modificarlas, agregar nuevas y/o darlas de bajas, entre otras funcionalidades. La Tesorería también puede seguir registrando, modificando y/o dando de baja cuentas. También se agrega la vinculación con el Sistema Integrado de Administración Financiera (e-Sidif). Esta comunicación agrega la posibilidad de contar con datos actualizados para la creación de cuentas, con esto se deja de tener datos duplicados en la base de datos de SIRECO, que luego quedan desactualizados.

1.3 Estructura de la tesis

En el Capítulo 1 se presenta la motivación por la cual se realiza la implementación, junto con el objetivo que se requiere cumplir con el nuevo aplicativo. Además, se realiza una breve introducción del surgimiento del Sistema de Registro de Cuentas Oficiales y su finalidad.

En relación con el Capítulo 2 se realiza la puesta en contexto de donde está situado el sistema, explicando la historia de la Organización y los distintos sistemas que fueron surgiendo durante el transcurso de estos años de una manera breve. A su vez, se expone las desventajas del sistema anterior y los comienzos del nuevo aplicativo dando las distintas ventajas que provee.

[1] e-Sidif: sistema que da soporte a la gestión presupuestaria, financiera, y contable del sector público nacional con el objetivo de proporcionar efectividad, eficiencia y transparencia del gasto público.

Por lo que se refiere al Capítulo 3 se da una introducción a las metodologías ágiles, tomando a Scrum como punto de partida del proceso de agilidad en el proyecto. Se sigue con una breve explicación de DevOps y Design Thinking. Además de dar una visión sobre el uso de Scrum y DevOps en conjunto.

En cuanto al capítulo 4 se realiza una explicación de la utilización de las distintas metodologías y herramientas (descriptas en el capitulo anterior) dentro del proceso para poder lograr un buen resultado final de la aplicación. Nombrando etapas, equipos, la interacción entre ellos, distintas reuniones con sus alcances. Además de los despliegues de las distintas entregas en los diferentes entornos y sus planificaciones.

Y por último en el Capítulo 5 se da una descripción del Sistema SIRECO, mostrando el propósito del mismo, junto con la migración de datos que se debió llevar a cabo para poder dar comienzo con el uso del sistema. A su vez, se incluye algunas definiciones destacadas dentro del aplicativo, como por ejemplo: Cuenta, Entidades Básicas, Constancia, entre otros.

1.4 Introducción

El Sistema de Registro de Cuentas Oficiales (SIRECO) surge a partir del cumplimiento del artículo 78, punto 5.1 del Anexo del Decreto 1344/2007[2], donde su principal objetivo es permitir llevar a cabo el registro de Cuentas Oficiales del Sector Público Nacional. Estas cuentas pertenecen a Empresas, Fondos Fiduciarios, Universidades, Otros Entes y Organismos de la Administración Pública Nacional.

Conjuntamente con la Disposición N° 5 donde la Tesorería General de la Nación (Tesorería) aprueba el *"Procedimiento para autorizar la apertura, modificación y cierre de Cuentas Oficiales" [3]*, lo que se trata de hacer con esta norma es facilitar el registro de las Cuentas Oficiales y mantener actualizados los datos del Registro de Cuentas Oficiales.

[2] El artículo 78, punto 5.1 del Anexo del Decreto 1344/2007, Reglamentario de la Ley 24.156 de Administración Financiera y de los Sistemas de Control del Sector Público Nacional, responsabiliza a la Tesorería General de la Nación (TGN) de mantener una base de datos en las que estén contenidas todas las cuentas bancarias autorizadas y/o informadas por los organismos del Sector Público Nacional. Según el Decreto anteriormente mencionado la base de datos es denominada Registro de Cuentas Oficiales, he aquí donde nace las siglas SIRECO (Sistema de Registro de Cuentas Oficiales).

Las cuentas pueden ser ingresadas por la Tesorería para un Organismo en particular o el Organismo solicitar la apertura de la misma. Esta apertura consiste en que el Organismo solicite el alta de la misma, en este paso es donde la Tesorería ingresa una cuenta sin número (con esto se autoriza al Organismo a realizar la apertura de cuenta en la entidad bancaria) y, una vez que el Organismo abrió la cuenta en el Banco, se completan los datos de la cuenta abierta en el sistema.

Capítulo 2

2.1 Un poco de historia

Antes de la creación de la Dirección General de Sistemas de Administración Financiera (DGSIAF), en la Secretaría de Hacienda, los sistemas eran desarrollados en entornos disímiles, con distintos alcances y niveles de seguridad, no había documentación que acompañe a las distintas aplicaciones; además de todo esto, estas no se adecuaban a la Ley de Administración Financiera 24.156. Con el nacimiento de la DGSIAF, en el año 1991, se le pudo dar un mejor cauce a estas aplicaciones, pudiendo así mejorarlas y crear nuevas.

Entre los años 1991 y 1995 se realizó una renovación del Sector Público, la cual da origen al Sistema Integrado de Información Financiera (SIDIF), el cual tenía como objetivo principal la formulación presupuestaria nacional y el registro de la ejecución presupuestaria. El SIDIF fue pensado inicialmente como un sistema integrado, compuesto por varios subsistemas y módulos dentro de un enfoque funcional, con una base de datos central (SIDIF Local AC), la cual estaba relacionaba con bases de datos locales de los Servicios de Administración Financiera (SAF) (SIDIF Local OD).

Sujeto a este orden, la Secretaría de Hacienda se conectaba con los sistemas locales y el resto de las aplicaciones a través del SIDIF Central. Las bases de datos locales guardaban la información de gestión y la base central agrupaba el registro de la ejecución presupuestaria.

Dentro de esta estructura coexistían una diversidad de sistemas locales en los distintos organismos cuyas características eran disímiles; esto generaba un incremento en el costo de mantenimiento y el retardo que implicaba replicar los ajustes que eran necesarios para los sistemas locales.

A causa de lo anterior mencionado, en el año 1999, se dio inicio al desarrollo de un nuevo aplicativo, cuyo objetivo era unificar los sistemas locales en un sistema único. En este se actualizaron y extendieron los requerimientos funcionales. Este nuevo sistema se llamó SLU (SIDIF Local Unificado). Este sistema trajo consigo una importante y gran mejora

en la gestión; suministrando información confiable y ajustando distintos procesos como el de compras, tesorería, presupuesto y contabilidad.

En el año 2004, luego de haber implantado una gran reforma a partir del SLU, incorporando mejoras significativas en la gestión de la administración financiera pública nacional, a partir de los avances tecnológicos y con el surgimiento de nuevos requerimientos funcionales se da comienzo a un nuevo desafío que se denominó e-Sidif.

El sistema propuesto busca mejorar las características del SIDIF incorporando nuevas tecnologías e incluir nuevos requerimientos y, además, incorpora transformaciones que favorecen una mayor gestión pública orientada a resultados ya que promueve vinculaciones con otros sistemas, sean internos a la DGSIAF u otros organismos.

E-Sidif cuenta con una base de datos centralizada, la cual contiene toda la información de la gestión financiera tanto de los servicios de administración financiera como de los órganos rectores.

Buscando federalizar el uso de la aplicación y responder a requerimientos y necesidades de otros organismos y administraciones, se comienza la adaptación e implementación del sistema e-Sidif para su uso en la gestión de provincias y entes autárquicos. La aplicación se ajusta a las exigencias nacionales, como así también a las de cada provincia o ente en la que se implemente.

Con el correr de los años y a partir del surgimiento de nuevas necesidades, se fueron desarrollando nuevos sistemas vinculados al e-Sidif destinados a diferentes usuarios (empresas públicas, ciudadanos, entre otros). Paralelamente se desarrollaron sistemas para usuarios internos de la administración pública nacional, que ayudan a la consulta, seguimiento y gestión de información.

2.2 ¿Cómo comenzó?

Como se ha mencionado anteriormente la Tesorería necesitaba una herramienta en la cual los Organismos pudieran registrar, validar, dar de baja cuentas. Esto es informar cualquier novedad que surgía sobre las mismas, de las cuales ellos eran los titulares.

Además, era necesario que esta entidad también pudiese realizar altas, bajas, modificaciones de cuentas.

Para entender cuál era la situación en ese momento, se realizaron varias reuniones con los usuarios de la Tesorería, los cuales fueron expresando las distintas dificultades con las que fueron encontrándose a lo largo de la utilización del sistema anterior. A su vez, fueron enumerando los distintos requerimientos que necesitaban en el nuevo sistema.

Desde la DGSIAF se le realizó una propuesta a la Tesorería, con el objetivo de resolver los requerimientos pedidos por la misma. En la que se proponía implementar una nueva aplicación web de Registro de Cuentas Oficiales, donde el titular de las cuentas pudiese ratificar el stock a su nombre y mantener al día los datos de las mismas. Así mismo, la Tesorería también pudiese realizar la administración de las cuentas, consultas, reportes variables que puedan responder a solicitudes judiciales entre otras operaciones.

Además, este proyecto agrega la comunicación con e-Sidif, el cual se utilizaba para el alta y actualizaciones de las cuentas operativas de los Organismos de la Administración Pública Nacional.

La decisión de la DGSIAF, desde hace unos años, es implementar las aplicaciones nuevas independientes del e-Sidif, esto es para desacoplar el comportamiento y no generar una aplicación monolítica. La idea de realizarla de forma independiente es poder desarrollarla y ponerla en marcha en forma incremental y desacoplada del e-Sidif.

Una vez aceptada la propuesta por parte de la Tesorería, se comenzaron con las reuniones entre los diferentes equipos de la DGSIAF, las cuales involucran recursos humanos de diseño y desarrollo, UX/UI, testing, análisis, CAU (centro de atención a usuarios), base de datos, etc.

De estas reuniones fueron surgiendo distintos temas, como por ejemplos la creación de la base de datos, tecnologías que se iban a utilizar para el desarrollo de la aplicación, distintas habilitaciones que se iban a necesitar para el desarrollo y la puesta en marcha del nuevo sistema. También se partieron de varias pautas, como por ejemplo la utilización OKD para el despliegue de la aplicación, las tecnologías para el desarrollo, Angular (luego con el transcurso de las reuniones se decide utilizar Angular Material), Java, etc.

A lo largo de estas reuniones se depuraron distintos objetivos y se armaron varias etapas de entregas con sus contenidos. Se llegó a realizar una planificación estimativa la cual fue presentada y aceptada por la Tesorería.

2.3 Desventajas del Sistema anterior utilizado para registrar cuentas

El usuario del Sistema (SIRECO), se encontró con varias dificultades, las cuales paso a detallar:

- El acceso al sistema era solo para la Tesorería.
- Procedimiento en el alta de una cuenta.
- Actualización de los datos mediante una planilla Excel.
- No había comunicación con el e-Sidif.

<u>El acceso al sistema era solo para la Tesorería</u>

La Tesorería General de la Nación era la responsable de cargar y modificar todas las cuentas, ya que el Organismos no tenía acceso al mismo por más que ellos sean los titulares de las cuentas. Trayendo sobrecarga de trabajo a la hora de realizar verificaciones y validaciones de las cuentas.

<u>Procedimiento en el alta de una cuenta</u>

Para dar de alta las Cuentas Oficiales en este sistema, existía un procedimiento donde se especificaba la información que debía ser presentada por el titular de la cuenta a la Tesorería.

- Un Organismo (titular) solicitaba la apertura de una cuenta a la Tesorería.
- Esta autorizaba la apertura de la cuenta e ingresaba los datos de la Cuenta en el Sistema SIRECO (Banco, Sucursal, entre otros).
- El Organismo iba al Banco y realizaba la apertura de la cuenta.

- Él mismo le informaba a la Tesorería que la cuenta se encontraba abierta en el Banco y le informaba los datos necesarios para dar de alta la cuenta en el sistema SIRECO, como por ejemplo el número de la misma.
- En este caso la Tesorería era la encargada de completar los datos de la nueva cuenta en el Sistema SIRECO. Con este último punto finalizaba el alta de una cuenta.

Actualización de los datos mediante una planilla Excel

Durante el año se habilitaba un período de tiempo, este se denomina Período de Empadronamiento, es en donde los Organismos estaban obligados a informar a la Tesorería el estado de las Cuentas Bancarias de las que eran titulares.

En reiteradas oportunidades, se observó que había cuentas que no se utilizaban y en este período eran dadas de baja y por otro lado, cuentas que no habían sido informadas a la Tesorería y el Organismo podía pedirle que las diera de alta.

Como los Organismos no tenían acceso al Sistema, la forma que se encontró para realizar la actualización de las cuentas por parte de estos, era que la Tesorería le proporcionaba una planilla Excel con la información de cuentas, esta plantilla incluía todas las cuentas de ese Organismo, el cual debía revisar y actualizar a mano en la planilla. Una vez que el titular chequeaba los datos de las cuentas, devolvía esta planilla a la Tesorería.

La verificación de datos de las cuentas mediante este proceso, el manejo de planillas Excel para datos tan sensibles, trajo muchas complicaciones ya que cuando recibían los datos de las cuentas actualizados de cada Organismo, los mismos tenían diferentes formatos y esto hizo que fuera más complejo la actualización de los datos de las cuentas para la Tesorería, donde llevaba mucho tiempo pasar toda esa información al Sistema de Registro de Cuentas Oficiales.

No había comunicación con el e-Sidif

Adicionalmente, las Cuentas Oficiales cuyos titulares eran Organismos de la Administración Pública Nacional y que se utilizaban en alguna operatoria de e-Sidif, la Tesorería las daba de alta en el Sistema e-Sidif de forma manual, lo que generaba una doble carga de datos. Esto se debía por la falta de comunicación con el Sistema e-Sidif.

Este sistema contaba, además de la consulta y actualización de Cuentas Oficiales, con las consultas y/o gestión de Bancos, Sucursales, Organismos, Tipo Cuenta, Caracteres, Clase, Fuente de Financiamiento, Monedas. En el Sistema estos datos están dentro de lo que se denominan Tablas Básicas.

Los datos de algunas de estas Tablas Básicas, como por ejemplo Bancos, Sucursales, Monedas y Fuente de Financiamiento, eran redundantes ya que estos datos también estaban en e-Sidif. El problema de esto es que si se realizaba alguna actualización en el e-Sidif de algún dato, no era transparente al Sistema SIRECO.

Al no permitir la comunicación con el e-Sidif, la actualización de los datos de las Tablas Básicas, que se nombraron anteriormente, se realizaba de forma manual, lo que esto llevaba a tener la información desfasada y/o distinta, ya que si se modificaba algo en el e-Sidif y no se realizaba dicha actualización en el Sistema SIRECO, este quedaba con información desactualizada, además de que los usuarios de la Tesorería podían realizar altas y modificaciones en estas Tablas Básicas, y estas operaciones podían llevar a que los datos queden diferentes a los que e-Sidif tenía. Esto podría dejar inconsistencia entre los sistemas.

2.4 Ventajas del Sistema actual para el registro de cuentas oficiales

Debido a las desventajas y/o dificultades que presentaba el sistema anterior (tales como la falta de comunicación con e-Sidif, el acceso era sólo para la Tesorería, entre otros) desde la DGSIAF (Dirección General de Sistemas Informáticos de Administración Financiera) se propuso el desarrollo de una nueva aplicación con el objetivo de mejorar los procesos utilizados en el aplicativo anterior.

Con esta nueva herramienta se logran mejoras los siguientes aspectos:

- Acceso al Sistema.
- Procedimiento en el alta de una cuenta.
- Chequeo y actualización de los datos de las cuentas.
- Comunicación con e-Sidif.

Acceso al Sistema

En esta nueva aplicación se incorpora a los usuarios de los distintos Organismos, donde se le da la posibilidad de corroborar los datos de sus cuentas, el stock a su nombre, y con esto poder tenerlas actualizadas. Dejando a la Tesorería General de la Nación con la absoluta facultad de poder administrar todas las cuentas.

Procedimiento en el alta de una cuenta

En la actualidad, este proceso ha agilizado en algunos puntos para la Tesorería, ya que ahora los titulares de las cuentas tienen acceso al sistema. A continuación se detallan los pasos necesarios para la apertura de una cuenta:

- El Organismo solicita la apertura de una cuenta a la Tesorería.
- La Tesorería lleva a cabo una Apertura de Cuenta en el sistema (Ingresando una cuenta con algunos datos necesarios como por ejemplo banco y sucursal; pero no ingresa el número de cuenta ya que aún no posee), y con esto el titular tiene la autorización para realizar la apertura en la entidad bancaria.
- Con esta autorización de apertura, el Organismo tiene que realizar los trámites necesarios en la entidad bancaria respectiva.

Hasta este punto es todo muy similar al sistema anterior, a partir del siguiente punto es donde se modifica el proceso de alta de una cuenta, agilizando el ingreso de nuevas cuentas.

- Luego de la apertura en la entidad bancaria, el Titular de la cuenta es el responsable de informar los datos, suministrados por la entidad bancaria, en el sistema SIRECO.

Chequeo y actualización de los datos de las cuentas

Una de las cosas más importante de este sistema es el alta de cuentas por parte de los Organismos, pero hay otro tema que no es menos importante, que se trata de la actualización de los datos de las cuentas por parte de los Organismos, se deja de lado la planilla Excel del sistema anterior, la cual traía diversos inconvenientes y pérdida de tiempo a la Tesorería tratando de actualizar todos los datos proporcionados por los titulares de las cuentas. Hoy en día, el sistema permite a los Titulares de las cuentas realizar la actualización de las mismas, dejando a la Tesorería la posibilidad de modificarlas también.

Comunicación con e-Sidif

Anteriormente, se comentó que el sistema anterior no contaba con la comunicación con el e-Sidif, esa conexión es importante ya sea para la obtención de distinta información relacionada con la cuenta, actualización de distintos datos de la misma, y a su vez tener la información al día de las Tablas Básicas.

En este punto, se logró tener comunicación con e-Sidif, ya sea para la creación y/o consulta de datos de la cuenta, como así también para tener los datos de las Tablas Básicas en consonancia con el mismo.

Capítulo 3

3.1 Metodologías ágiles

Las metodologías ágiles se utilizan en la gestión de proyectos (cada vez más dinámicos y variables) donde requieren de una flexibilidad y rapidez en adecuarlos, tratando de tener mínimas consecuencias, sin dejar de lado la calidad del producto. Se basan en el desarrollo iterativo e incremental, esto es agregando nuevas funcionalidades a la aplicación con cada iteración donde lo que se suma al producto es de importancia para el usuario.

Entre otras características de las metodologías ágiles se pueden encontrar:

- El equipo está capacitado para tener una autonomía, poder realizar distintas funciones y sobre todo auto-organizarse.
- El equipo busca mejorar la eficiencia y se adecua para alcanzarla.
- Favorecer la comunicación cara a cara que tener documentación excesiva.
- Soporta sin dificultad el cambio de requerimientos.
- Las entregas se realizan con una regularidad entre 1 a 4 semanas.

La gestión de proyectos ágiles pretende entregar en periodos cortos de tiempos partecitas de la aplicación en marcha para lograr que el usuario esté conforme y satisfecho. El agilísimo es: rápido, pronto, ya, ahora. Principalmente es lo que se requiere en un proyecto para impedir que no perdure eternamente.

3.1.1 Manifiesto por el desarrollo ágil del software

Este documento[5] fue elaborado en febrero del 2001 por 17 experimentados en programación donde se proponía un vuelco drástico en la manera de desarrollar aplicaciones. Para obtener un producto eficiente se deben poner en práctica los cuatro valores y doce principios que establecen las metodologías ágiles.

Los valores de agilidad que se encuentran en el manifiesto apoyan un cambio de mentalidad dejando de lado formas de trabajo:

- Reconocimiento de las personas y sus vínculos sobre los procesos y herramientas.
- Cooperar con los usuarios para alimentar una relación más cercana. Los usuarios son considerados copartícipes.
- En vez de tener documentación detallada se antepone el buen funcionamiento de cada entrega de la aplicación de forma incremental.
- Anticiparse y adaptarse a los cambios constantes en vez de continuar con un plan.

Este documento enumera doce principios en los cuales se centran las metodologías ágiles:

1. La prioridad es satisfacer al usuario con el producto por medio de distintas entregas.
2. Las distintas modificaciones que puedan sufrir los requerimientos se aceptan.
3. Las entregas se llevan a cabo de manera continua y el tiempo entre entrega y entrega son muy cortos.
4. El equipo debe trabajar diariamente de forma coordinada y colaborativa.
5. Se necesita que las personas que realizan el proyecto estén motivadas.
6. La comunicación cara a cara es más productiva que tener documentación excesiva.
7. Con el producto funcionando se ve el avance y los logros obtenidos.
8. Fomentar regularidad de trabajo continuo.
9. Durante todo el desarrollo del proyecto se debe buscar y tener en cuenta la excelencia técnica.
10. Lo sencillo es fundamental (entregas constantes, no perder tiempo y que el producto sea bueno).
11. Para poder realizar un buen producto es necesario que el equipo trabaje en conjunto y se auto-organice para lograr los objetivos propuestos.
12. Analizar en conjunto con todo el equipo lo que se necesita optimizar y así lograr mejoras continuas en las próximas iteraciones.

3.1.2 ¿Qué es Scrum?

Scrum[6] es una metodología ágil de gestión de proyectos, la cual se centra en las buenas prácticas y trabajo en equipo, donde el equipo es auto-organizado. Tiene un ciclo de

vida iterativo e incremental, esto hace que se potencie la productividad. Scrum se basa en tareas definidas previamente y en la división de etapas con distintas metas a cumplir. Esta metodología se adapta mejor en los proyectos donde los requerimientos cambian constantemente y se necesitan resultados rápidos.

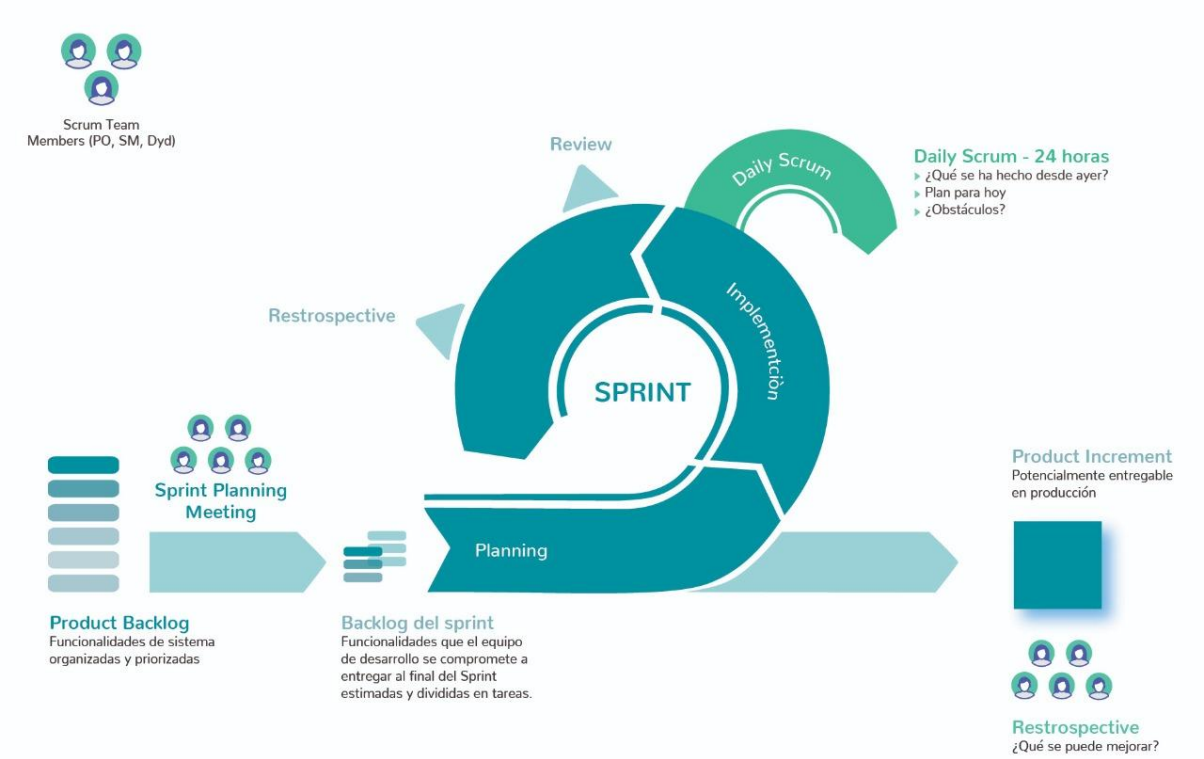

Figura 1 – Scrum

3.1.2.1 El contexto de trabajo de Scrum

Pilares fundamentales

- **Transparencia**

Las características relevantes del proceso deben estar expuestas para todo el equipo de trabajo. Todos deben ver lo que está pasando. Ejemplos:

- Planificación: se sabe lo que se va a realizar durante el sprint
- Daily: se ve lo que se hizo, lo que se va a realizar y si hay algún impedimento para la realización de alguna tarea.

- **Inspección**

  Es necesario revisar los artefactos (Product backlog, Sprint backlog, incremento), los avances realizados para alcanzar la meta. Esta revisión debe ser realizada frecuentemente para poder detectar desviaciones de los objetivos planteados, y así poder adaptar el plan y poder reorganizar las tareas para poder mejorar los resultados.

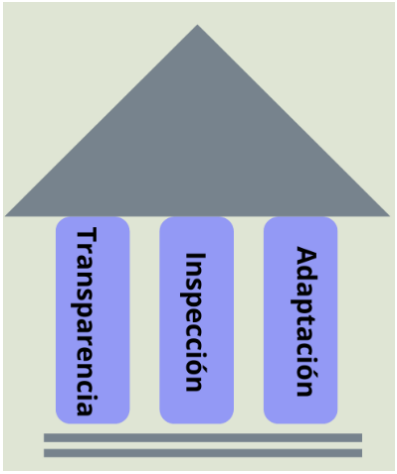

Figura 2 – Pilares de Scrum

- **Adaptación:**

  Durante la inspección pueden surgir cambios, estos puede ser que cambió algún requerimiento de usuario o hubo alguna desviación con respecto a los objetivos. El equipo se adecúa para poder lograr el objetivo.

Estos tres pilares están fijados sobre el empirismo, el cual se fundamenta en el conocimiento que se va adquiriendo de distintas vivencias y de realizar elecciones teniendo en cuenta lo conocido (lo que ya pasó).

Roles

- **Product Owner:**
  - Es la persona que mantiene contacto con el usuario por lo que tiene los conocimientos de lo que éste necesita.

- Es la encargada de alimentar el Product Backlog redactando las ideas del usuario y poniendo prioridades a cada una.
- Por cada equipo scrum hay un solo PO.

- **Scrum Master:**

  - Es el facilitador del equipo, ayudando a entender scrum al resto del equipo para que se aplique y obtener una alta productividad.
  - Líder que ayuda a resolver obstáculos o complicaciones que puedan surgir en el equipo.
  - Frecuentemente pertenece al equipo de desarrollo.
  - Garantiza la transparencia.

- **Equipo de desarrollo:**

  - Responsables de estimar y llevar a cabo las tareas del product backlog armando los sprints backlog para poder generar una entrega incremental de calidad.
  - Es un equipo auto-organizado y pueden realizar múltiples tareas.

Las personas que tienen estos roles pertenecen al Scrum Team, y están directamente relacionadas con el proceso scrum donde ordenan sus propias tareas para poder realizar una entrega incremental que contenga gran valor y beneficios para el usuario con cada finalización del sprint.

Además se encuentran los roles de Usuario (usa el producto), Stakeholders (el proyecto lo beneficia en algún aspecto) y Managers. Estos roles no están incluidos en el proceso de scrum pero son importantes para poder realizar un feedback de las entregas incrementales e iterativas.

Eventos

- **Sprint:**

  - Es el evento principal de scrum, el resto de los eventos transcurren dentro del sprint al igual que el resto de las actividades.

- La duración es entre 15 y 30 días, ya que si se hace más extenso se puede perder el objetivo del sprint.
- Al final de cada sprint se dispone un incremento del producto.
- Si se pierde el objetivo del sprint, el Product Owner es el único que puede decidir cancelar el sprint.
- Los sprints son consecutivos, termina uno y empieza otro.
- Cada sprint es una iteración sobre el producto.

- **Sprint planning meeting:**

  - Es la reunión que se realiza al comienzo de cada iteración donde se establece el objetivo del sprint.
  - A esta reunión concurre todo el equipo, el cual llegará a un acuerdo que tareas se pueden realizar durante el próximo sprint.
  - Las tareas se sacan del product backlog y se genera el sprint backlog. Cada tarea tiene su prioridad.

- **Daily meeting**

  - Reunión diaria de no más de 15 minutos donde cada desarrollador comenta al resto el avance de su tarea, para que el equipo pueda llevar a cabo un control y de ser necesario, realizar una adaptación para poder cumplir con el objetivo del sprint.
  - Hay tres preguntas que se hacen en cada daily:

    - ¿Qué se hizo?
    - ¿Qué se va a hacer?
    - ¿Hay alguna imposibilidad para poder realizar la tarea?

- **Sprint review**

  - Reunión que se realiza con cada finalización del sprint donde se muestra lo nuevo desarrollado.
  - Además del equipo de scrum, también participan las personas interesadas (usuario, stakeholders, etc.)

- El usuario realiza un feedback con respecto a lo entregado lo cual puede llevar a un ajuste en el product backlog para poder realizar una adaptación de lo que el usuario observó.

- **Sprint retrospective**

  - Asiste el equipo de desarrollo, product owner y el scrum master.
  - Esta reunión da por finalizado el sprint, en la cual se plantean qué cosas se hicieron bien, cuales se pueden mejorar para poder tener mayor productividad en futuros sprints.

Artefactos

- **Product backlog**

  - Es una lista de requerimientos que el sistema debe cumplir según lo expresado por la parte interesada.
  - Esta lista es armada, actualizada y priorizada por el product owner.
  - Cada uno de los requerimientos son estimados por los desarrolladores.

- **Sprint backlog**

  - Lista de tareas que se alimenta del product backlog y se crea en la planificación de cada sprint.
  - Un requerimiento del product backlog puede ser dividido en varias tareas por el equipo de desarrollo.
  - Esta lista no se puede modificar durante el sprint, el equipo de desarrollo es el único que puede agregar o sacar tareas durante el sprint, por lo general ocurre cuando la planificación se realizó de manera errónea.
  - A cada tarea se le asigna una prioridad y puede depender de otras tareas.
  - Es armada por el equipo de desarrollo con ayuda del product owner.

- **Incremento**

    - Lista de requerimientos entregados en una iteración y pueden ser utilizadas por el usuario.

Valores

- **Compromiso:** el equipo de desarrollo se compromete para poder alcanzar el objetivo.
- **Respeto:** los desarrolladores se respetan entre sí.
- **Coraje:** el scrum team tiene que tomar decisiones difíciles para poder solucionar determinados problemas.
- **Franqueza:** cada integrante del scrum team debe ser transparente hacia el resto del equipo, tanto con el trabajo realizado como con el conocimiento adquirido.
- **Foco:** El equipo de desarrollo se centra en el objetivo del sprint para poder obtener los resultados esperados.

La utilización de Scrum brinda alguna de estas ventajas:

- Los usuarios pueden colaborar con el desarrollo y dar opiniones sobre posibles soluciones.
- Los usuarios no tienen que esperar a la finalización del producto para poder ver los resultados, sino que con cada entrega incremental se ven los requerimientos entregados.
- Se adapta a los cambios, ante cualquier eventualidad el equipo se coordina y se adapta para poder solucionarlo.

3.2 DevOps

¿Qué es DevOps?

DevOps[7][8] se trata de un enfoque que se utiliza para la creación de sistemas de forma ágil, rápida, eficiente, a bajo costo y automatizada. La idea de este movimiento cultural es acercar el equipo de desarrollo (Dev - trabaja para realizar mejoras en el sistema

en menor tiempo adaptándose al cambio constante) y el equipo de operaciones (Ops - es el que pone en marcha, optimiza y da seguridad el sistema en los distintos ambientes), donde estos equipos se comunican, coordinan y trabajan en conjunto para entregar un producto con un gran valor para usuario/cliente de forma incremental.

DevOps se relaciona con el desarrollo ágil, la nube y la automatización de ciertas operaciones para poder realizar integraciones y entregas continuamente. De esta forma, se obtiene mayor flexibilidad, incremento en la productividad, mayor calidad y despliegues continuos. La idea es realizar despliegues con regularidad en periodos de tiempos cortos, esto hace que haya un incremento en la productividad ya que con cada entrega se suma funcionalidad al sistema obteniendo una mejora en la calidad del sistema disminuyendo la cantidad de errores (no es lo mismo entregar cada 15 días que hacerlo cada 4 meses).

Elementos

Existen varios elementos dentro del desarrollo ágil de DevOps:

- ***Control de versiones***: da la posibilidad de almacenar en un repositorio el código y visualizar los distintos cambios realizados en el proyecto, se pueden tener varias ramas donde cada una pueda ser utilizada para alguna funcionalidad en particular. Ejemplo Git es la herramienta para el control de versiones y GitLab es el repositorio para el control de versiones.

- ***Gestión de entornos*** (OpenShift): crear distintas etapas por donde la entrega va a ir transitando (desarrollo, prueba y de producción donde cada uno son diferentes). Para que no haya errores de entornos se genera una virtualización del desarrollo y se pasa a testing; se promueve a producción cuando se culmina con todos los testeos exitosos.

- ***Automatización de la configuración del sistema:*** al realizar un cambio en algún entorno se debe copiar en el resto de los entornos, para esto se utilizan distintas herramientas que realizan de forma casi automática estos cambios en los distintos entornos.

- ***Testeos***: realizar pruebas unitarias de código, prueba de porciones de código; y con las pruebas funcionales, probar el todo.

- ***Integración***: utilizar herramientas para automatizar la integración continua y pruebas continuas (jenkings).

- ***Comunicación***: fomentar la comunicación fluida para que todos puedan sacarse las dudas que puedan surgir y entiendan lo que hay que realizar. Chat (Mattermost), correos, etc.

Etapas

DevOps se modela como un bucle infinito, este bucle representa el proceso continuo de integración y de entregas continuas. A continuación se describen las fases que componen el bucle de DevOps:

1. ***Planificación***
   Se reúnen los equipos de desarrollo, operaciones y otras personas interesadas para definir los objetivos y las tareas de cada equipo dentro de la iteración.

2. ***Desarrollo/construcción***
   El equipo de desarrollo da soluciones a los objetivos planteados en la planificación, comprobando que la solución es la correcta, el código escrito se guarda en un repositorio general. Luego este código se compila, se realizan pruebas y se publica en un repositorio compartido.

3. ***Pruebas***
   Se realizan pruebas para cada parte implementada en la etapa anterior, estas pruebas son automatizadas y definidas con anticipación. Esto se realiza para poder detectar errores tempranamente.
   Tener pruebas automatizadas no quiere decir que se sustituye el testeo o las pruebas manuales, sino que es algo más que ayuda a la detección de errores.

4. ***Empaquetar***
   Se empaqueta el código para luego llevarlo a cualquier ambiente evitando cualquier posible error.

5. ***Despliegue***

   DevOps suprime el despliegue manual con cada entrega de desarrollo automatizando las entregas y agilizando la puesta en producción. Cada vez que se termina de implementar un requerimiento se realiza el despliegue. (Despliegue continuo).

6. ***Configuración***

   En esta fase se configura el sistema y la infraestructura de manera dinámica, según la aplicación lo necesite. El objetivo es proteger información sensible, mejorar el rendimiento, procesos y procedimientos.

7. ***Monitoreo***

   Se analizan las métricas del desempeño del código y performance en producción, esto es para poder optimizar el sistema y dar una mejor calidad en entregas futuras.

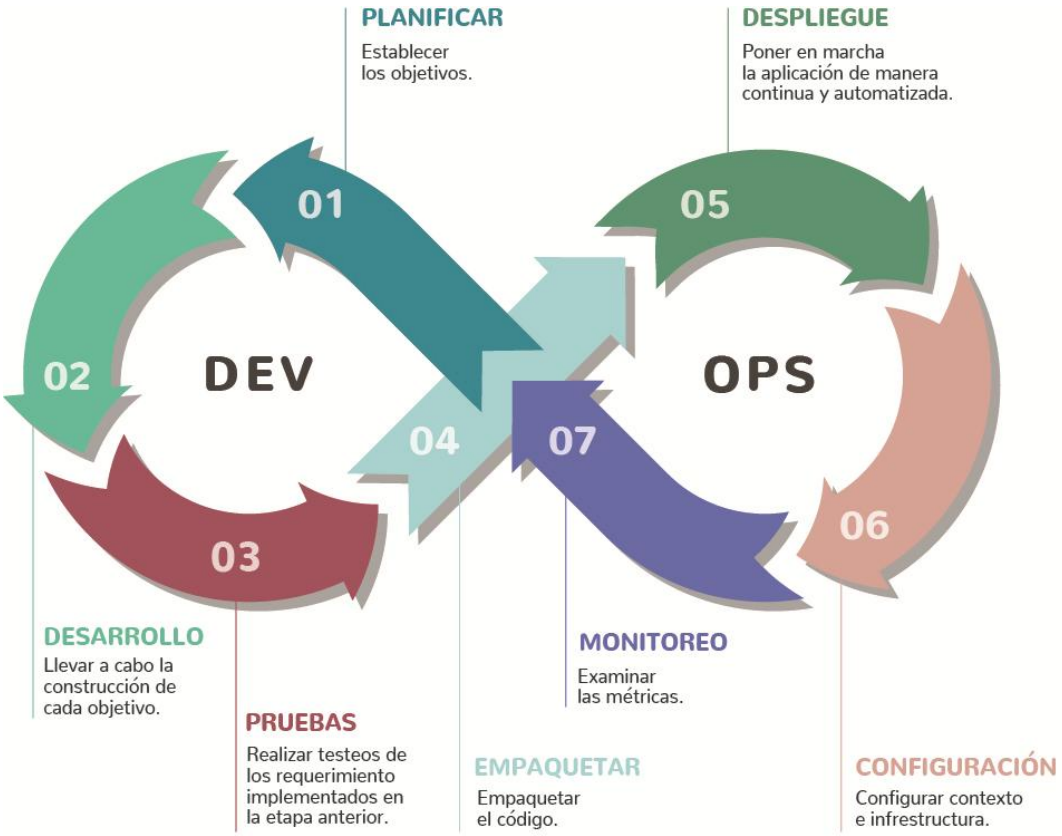

Figura 3 – Fases de DevOps

3.3 Utilizar Scrum y DevOps al mismo tiempo

DevOps fomenta la integración y el despliegue continuo de forma automatizada de las aplicaciones, facilitando mejorar la calidad y el rendimiento de manera más rápida en las aplicaciones. Además de hacer foco en la mejora de la eficiencia en el desarrollo, implementación y el mantenimiento de la aplicación. Scrum, ante los cambios frecuentes de los requerimientos, tiene la cualidad de adaptarse de manera rápida, pone énfasis en aligerar el desarrollo dejando de lado los testeos. Cosa que DevOps tiene muy presente.

La utilización de scrum es útil a la hora de realizar el acompañamiento de los distintos requerimientos planificados. Muchas veces los requerimientos de distintas operaciones no pueden esperar a ser priorizadas y planificadas en un sprint próximo, ya que requieren una reacción rápida para su solución (Ejemplo: error en la seguridad). DevOps atiende estos requerimientos tan pronto como surgen, además realiza la revisión del código, pruebas automatizadas, la integración y despliegues continuos.

La utilización de DevOps y Scrum en forma conjunta favorece al desarrollo y a la implementación de aplicaciones con una marcha mucho rápida. Donde Scrum apunta al equipo de desarrollo, en tanto que DevOps apunta a los grupos que forman parte del desarrollo y de las operaciones.

3.4 Design Thinking

Design Thinking[9] es una metodología dentro de la experiencia de usuario, que ayuda a la resolución de problemas dando distintas opciones innovadoras e ingeniosas de forma colaborativa, donde siempre se pone en el centro al usuario y sus necesidades teniendo que sea tecnológicamente factible y económicamente posible.

Es un proceso no lineal, es decir, que se puede regresar a fases anteriores, avanzar u omitir otras para poder rearmar como también replantear la solución. Se busca obtener el mejor resultado donde se cumpla con las necesidades del usuario. También es iterativo y está compuesto por cinco fases: Empatizar, Definir, Idear, Prototipar y Testear.

Fases

1. ***Empatizar***

   La idea de esta fase es ponerse en el lugar del usuario dejando todo tipo de conjeturas de lado para poder entender y conocer las necesidades del usuario y el contexto en el cual se encuentra. Esto se logra con reuniones y distintas técnicas de UX.

2. ***Definir***

   De la información obtenida en la fase anterior se extrae lo más relevante para poder enmarcar los principales problemas que el usuario enfrenta y de esta manera se continúa con el proceso de diseño para lograr encontrar una solución innovadora.

3. ***Idear***

   El objetivo de esta fase es explorar alternativas y/o posibles soluciones para un mismo problema, es decir, que a través del pensamiento divergente se observan todas las ideas que surjan sin descartar ninguna de ellas, utilizando distintas técnicas de brainstorming en los talleres colaborativos. Luego, se aplica el pensamiento convergente, de todas las ideas planteadas se analizan cual/es son la/s mejor/es entre todas para continuar con el proceso.

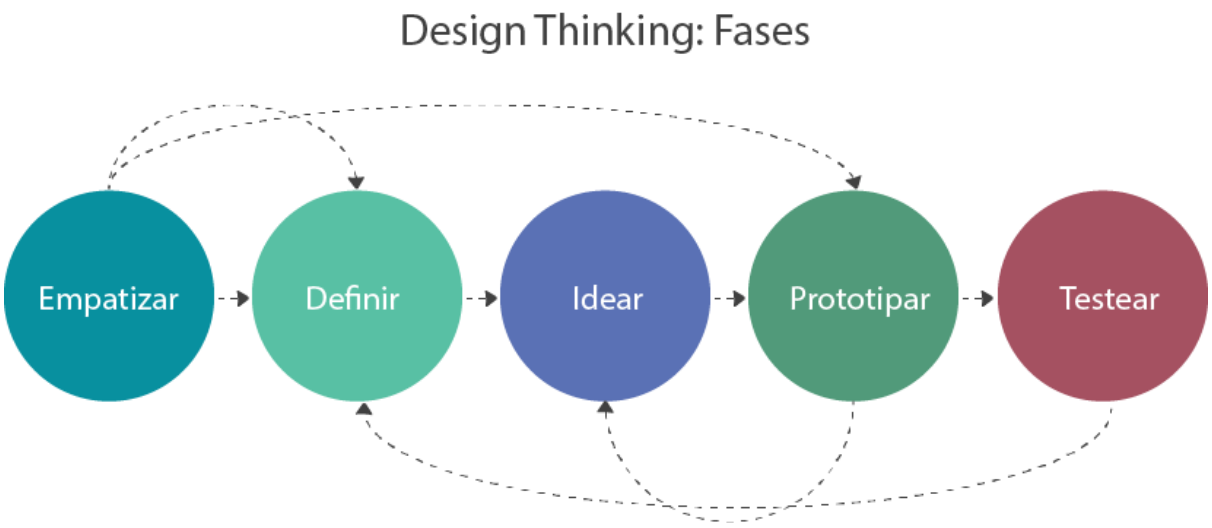

Figura 4 – Fases de Design Thinking

4. ***Prototipar***

   Con las ideas propuestas en la fase anterior se construyen distintos prototipos, donde se pueden realizar físicamente (en papel, plastilina, etc.) o digitalmente,

siempre y cuando la construcción no sea costosa. En los prototipos se pueden ver las ventajas y las mejoras a realizar para obtener la solución más adecuada.

5. ***Testear***

   En esta fase el usuario prueba los prototipos y realiza distintas devoluciones donde se detectan puntos de dolor, mejoras y caminos alternativos, con información detallada en un informe y compartida con el equipo del proyecto, se reajustan los prototipos para realizar una próxima iteración.

Capítulo 4

4.1 Agilidad aplicada en el proyecto

Como antes ya se mencionó este proyecto se realizó desde cero no solo el análisis, prototipado, diseño, desarrollo y testeo; sino que también se agregó una parte importante, que es la participación constante de los usuarios desde el comienzo, haciéndolos participes desde la construcción del prototipo hasta las pruebas en ambientes de aceptación (staging). Además de la incorporación de herramientas para la organización de las tareas, documentación de las distintas especificaciones de los requerimientos solicitados por los usuarios y desarrollo de la aplicación.

4.1.1 Etapas del proyecto

Previamente se comentó que el desarrollo y la puesta en producción de esta aplicación serán de forma iterativa e incremental. Para esto se planificaron distintas etapas donde cada una estaba compuesta de distintos objetivos a cumplir. Antes de comenzar con el desarrollo del aplicativo se llevó a cabo la pre-etapa I donde:

Pre Etapa I

Capacitación a los distintos equipos en las diferentes herramientas que se utilizaron (GitLab, OKD, Angular, Material, etc.). A su vez, el equipo de innovación elaboró recomendaciones metodológicas para el equipo de SIRECO, las cuales ayudaron a ordenar y fijar los distintos objetivos de cada equipo.

Luego se llevaron a cabo las etapas donde se analizó, prototipo, desarrolló y testeó el sistema:

Etapa I

Se da comienzo con el análisis funcional de la aplicación, manteniendo distintas reuniones con los usuarios.

Se incorporó el equipo de Diseño UX/UI a la dinámica del equipo de SIRECO, donde colaboró con el prototipado, en las reuniones intra equipo para las validaciones de las distintas definiciones por medio del diseño del prototipo y validaciones de diseño con los usuarios.

Se desarrollaron los siguientes requerimientos:

- Funcionalidad para que la Tesorería pueda realizar distintas operaciones sobre las cuentas.
- Migración de las cuentas oficiales desde el sistema anterior.
- Consulta de las tablas básicas, (estas tablas contienen datos que son utilizados para la gestión y/o consulta de las cuentas).
- Migración de datos de las tablas básicas específicas del aplicativo.
- Comunicación con e-Sidif para consultar datos para algunas tablas básicas.
- Reporte dinámico de las cuentas (Formato XLSX y PDF).
- Consulta de cuentas detallada, la cual está compuesta por una diversidad de filtros, los cuales ayudan a buscar información variada.
- Autenticación y autorización para los usuarios de la Tesorería.
- Historia para las modificaciones de las cuentas.
- Cron con el cálculo de estados de las cuentas.

Etapa II

- Revisar cuentas por un organismo.
- Revertir revisión de cuenta para un organismo.
- Revisar cuentas por la Tesorería.
- Revertir revisión de cuenta para la Tesorería.
- Generar constancia: El organismo luego de revisar todas las cuentas genera una constancia.
- Periodo de empadronamiento: período durante el cual un organismo puede modificar, dar de baja y/o crear cuentas.
- Autenticación y autorización para los usuarios de organismos que se loguean con el Cuit, se valida contra un web service de AFIP.
- Autenticación y autorización para los usuarios de organismos que se loguean con el DNI, se valida contra e-Sidif.

4.1.2 Planificación

Se puede decir que se realizaron distintas planificaciones a corto, mediano y largo plazo. Dividiendo los requerimientos que los usuarios necesitaban para las distintas etapas de entregas.

En la planificación a mediano y corto plazo se utilizó la herramienta GitLab[11], en la cual se realizaba el seguimiento de las distintas tareas. Se armó un tablero con distintos carriles donde cada tarea (issue) podía ir pasando.

Al principio, las reuniones de planificación se realizaban semanalmente, luego de tener avanzada la aplicación se decidió realizar la planificación a corto plazo cada un período de 15 días (equivalen a un sprint). Para este período de tiempo se planificaba lo que el equipo de desarrollo podía implementar y por otro lado, qué testing alcanzaría a probar en el siguiente sprint. Las prioridades de cada tarea que se incluye en estos sprint fue puesta por el equipo de análisis (revisando cada tarea, le ponía un peso a cada una para saber cuál es la más urgente o la que tenía más prioridad con respecto a las demás).

Por otro lado, se tenía la planificación a largo plazo, para esta parte se decidió tener una planilla (Figura 5) compartida entre los distintos equipos. Esta planilla listaba todos los temas que surgieron, con la posible fecha estimativa que podían ser tomadas por el equipo de desarrollo y testing, a su vez tenían la estimación de cada tarea y prioridades.

De cada tarea se podía visualizar:

- La estimación tanto lo que llevaba el desarrollo como el testeo de cada tarea.
- Si la tarea estaba terminada o planificada para que la tomara desarrollo.
- Si se había generado la entrega y en cual salió.
- Se podía ver el número de issue, el cual hace referencia al GitLab.

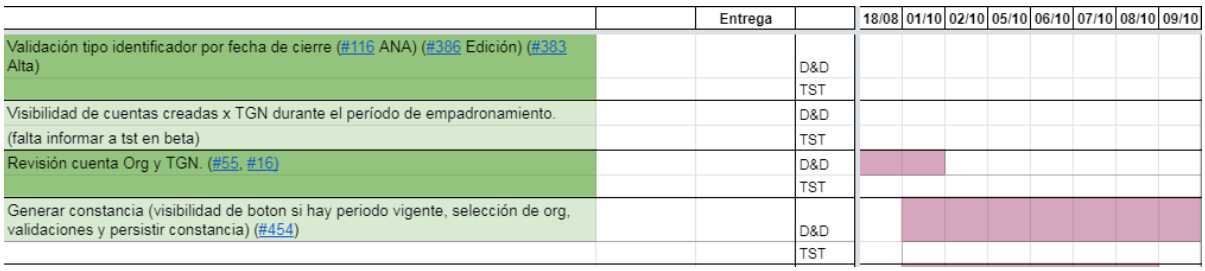

Figura 5 – Planificación largo plazo

Esta planificación a largo plazo se revisaba constantemente para ver si era necesario incluir alguna de esas tareas dentro de las planificaciones de los próximos sprints o agregar nuevas tareas para su posterior planificación.

4.1.3 Sprints

Al principio del proyecto se optó por sprints semanales, y luego se pasaron a sprints quincenales. En las reuniones de planificación se ponían los objetivos para el sprint siguiente y se observaba de algunas tareas posibles desviaciones o requerían intervención de grupos/personas externas al proyecto. En la finalización de cada sprint, el equipo de desarrollo generaba una entrega programada (contenía todas las tareas finalizadas en su desarrollo).

4.1.4 Reuniones

Además de las reuniones de planificación (semanal o quincenal) se realizaban reuniones semanales donde se hablaban temas más globales entre los equipos de desarrollo, testing, UX/UI, análisis, OKD, CAU:

- Pendientes.
- Lo que surgía durante el sprint.
- Los planteados por el usuario.
- Configuraciones que se debían realizar.
- Cada grupo exponía donde se encontraba parado.
- Estimaciones de requerimientos.

Cuando un tema ameritaba se llevaban a cabo distintas reuniones para llegar a una solución adecuada. Estando todos los grupos involucrados se ponían todas las ideas sobre la mesa y los puntos de vista de cada uno y esto ayudaba a llegar a una mejor solución.

Otras reuniones que se realizaban fueron con los usuarios, hubo una comunicación constante entre los usuarios, análisis, CAU y UX/UI. Con estas reuniones se recopilaba información de lo que el usuario iba viendo en la aplicación. O de surgir alguna duda o un nuevo requerimiento.

De cada reunión se generaba un documento (minuta) con la información de los participantes, temas tratados, alertas, etc. (Toda información relevante que pudiera servir en un futuro).

4.1.5 Prototipado

El equipo de UX/UI fue el encargado de llevar adelante el prototipado. Se realizaron distintas reuniones con los usuarios donde interactuaron con el mismo. A partir de estas pruebas se detectaron ajustes y mejoras las cuales se realizaron en distintas iteraciones.

El equipo de UX con el trabajo colaborativo de los distintos equipos, profundizó la tercera etapa de la metodología Design Thinking, “Prototipado”, realizando las iteraciones necesarias para avanzar durante el proceso. A medida que el prototipo evolucionaba en sus distintas fidelidades se fue incorporando distintas instancias de caso de uso. Luego, se incorporó las definiciones de color, tipografía, tamaño, entre otros.

Se comenzaron con prototipos en papel, posteriormente en distintas reuniones entre UX/UI y el equipo de análisis, en prototipos/wireframe en papel se plasmaron en un prototipo digital en fidelidad media (blanco y negro) con el contenido correspondiente a las interfaces (esto era para poder enfocar en la funcionalidad y la interacción usuario prototipo y no en colores y letras). Una vez que el prototipo quedó estable se incorporó las definiciones de color, tipografía, tamaño, entre otros (fidelidad alta). El prototipado digital se realizó con la herramienta Adobe XD.

4.1.6 Entregas

Nombres de las entregas

Cabe aclarar que los nombres de las entregas se definieron semánticamente dependiendo del alcance de cada una. Este alcance se basaba de hasta dónde podía llegar cada entrega a ser instalada (desarrollo, testing, staging, producción).

La nomenclatura de cada nombre se definió dentro del equipo de trabajo al comienzo del proyecto, este acuerdo de nombres lleva a un orden no solo para el equipo en

sí, sino que además para personas externas al proyecto. Cada nombre de una entrega es visto como una versión.

El nombre de la versión se podía ver como un conjunto de letras (X,Y,Z y W). Estas letras tenían un significado por la posición que ocupa:

- **X:** Versión importante (major), esta era incrementada cuando el contenido de la entrega era incompatible con versiones anteriores, refactors, migraciones, etc.
- **Y:** Una característica nueva a la versión actual,
- **Z:** Arreglos de bugs.
- **[W]:** se utilizaba para las entregas de testing y staging donde se iban sumando distintas funcionalidades, arreglos de bugs, etc.

A continuación se detallan los distintos nombres con los que se organizaban las distintas versiones de la aplicación:

- **X.Y.Z-beta.W**: Esta entrega se instalaba en el ambiente de testing y muere en ese ambiente.
- **X.Y.Z-rc.W**: Las entregas que tenían esta nomenclatura pueden ser instaladas hasta el ambiente de staging pasando por el de testing previamente.
- **X.Y.Z**: Estas entregas eran las que llegaban hasta el ambiente de producción, previamente atravesando los ambientes de testing y staging.

Todas las entregas anteriormente mencionadas quedaban instaladas en desarrollo antes de continuar el trayecto al ambiente destino.

A parte de las programadas existían las entregas que debían salir de forma urgente por algún error bloqueante, estas no llevaban ninguna programación de fechas, el equipo de desarrollo tomaba el requerimiento, antes priorizado por el equipo de análisis. Se realizaba la entrega cuando el equipo de desarrollo lo solucionaba y el equipo de testing lo podía probar.

Avanzado el desarrollo de la aplicación surgió la necesidad de modificar la base de datos (update, creación, eliminación de datos) sin entregar contenido de código nuevo, esto se debe a la corrección de datos por pedidos de los usuarios. Estas entregas se ejecutan con el Job de liquibase que se configuró en OKD.

A continuación se detalla el proceso que se definió para la generación de entregas solo SQL sin importar que sí había código nuevo en el branch desde donde es realizada la entrega, sólo se tenía en cuenta que desde el lugar que se realizaba la entrega no debía haber ninguna diferencia a nivel de SQL con el de producción (sólo debe existir el “cocinero” que se quiere ejecutar), esto se debe a que Liquibase corre todo lo que no está corrido en la base donde se está ejecutando.

El nombre para cada archivo SQL se acordó que contenga una identificación de lo que realiza sumado redmine y el número de la petición (si es que está relacionado a una petición redmine). Ejemplo: 02_INSERT_ORG_REDMINE_160.xml, esto quiere decir que se insertaron organismos realizados en la petición #160 mediante el redmine.

Proceso que se definió para realizar una entrega de modificación de datos:

- Verificación que no haya diferencias entre el ambiente donde se genera la entrega y de producción.
- Generar el SQL/XML con lo que se requiere modificar en la base de datos.
- Por única vez, crear una carpeta “cocinero” (nombre amigable) y agregar al db.changelog.xml principal.

```
<!-- COCINEROS-->
<include relativeToChangelogFile="true" file="cocineros/db.changelog.xml" />
```

Figura 6 – Archivo de llamada de los cocineros

- Agregar el XML (en esta aplicación se optó por realizar XML en vez de SQL) a la carpeta destinada para los distintos cocineros.

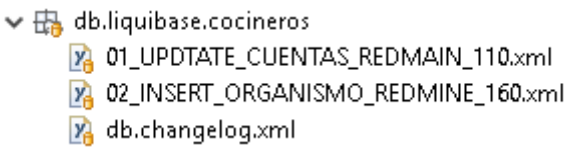

Figura 7 – Estructura de la carpeta

- Agregar cocinero en el db.changelog.xml ubicado dentro de la carpeta “cocineros” y pushear los cambios en el branch.

```
<!-- aca se incluyen los cocineros -->
<include file="01_UPDTATE_CUENTAS_REDMAIN_110.xml" relativeToChangelogFile="true" />
<include file="02_INSERT_ORGANISMO_REDMINE_160.xml" relativeToChangelogFile="true" />
```

Figura 8 – Llamada de cada cocinero

- Luego desde el GitLab, taguear el branch con algún nombre significativo para el cocinero que se entrega: **“COCINERO_02_INSERT_ORG_REDMINE_160”**
- Luego de la creación del SQL/XML ejecutar, para esto es necesario ingresar al OKD del ambiente de producción y crear un Liquibase Job. Y completar el GIT_REF con el nombre del tag del cocinero luego “create”, con esto se genera un POD que solo corre el cocinero.

Continuidad de entregas

Para una mejor organización dentro del equipo de trabajo se acordó un lapso de tiempo para cada tipo de entrega. Siempre hablando de las entregas programadas:

- Las destinadas a testing (**X.Y.Z-beta.W**) se realizaban con la finalización de cada sprint (a veces eran quincenales, otras semanales).

- Las destinadas a staging (**X.Y.Z-rc.W**) se realizaban a partir de la salida de funcionalidad nueva e importante para el usuario, acordando previamente en qué momento se realizaba entre el usuario y el equipo de trabajo. Estas entregas por lo general tenían una suma de funcionalidades entregadas en distintas **X.Y.Z-beta.W**. Se acordó no sacar funcionalidad nueva dentro de una **X.Y.Z-rc.W**, aunque el equipo de testeo realice pruebas para confirmar que la aplicación funcionaba correctamente.

- Con las entregas enviadas a producción (**X.Y.Z**) se resolvió realizarlas cada 3 semanas. Estas entregas podían contener un conjunto de **X.Y.Z-beta.W y/o X.Y.Z-rc.W**. Esto dependía de qué contenido se acordó entregar.

Documentación del contenido de cada entrega

El contenido de cada entrega se documentaba en el changelog.md del branch principal.

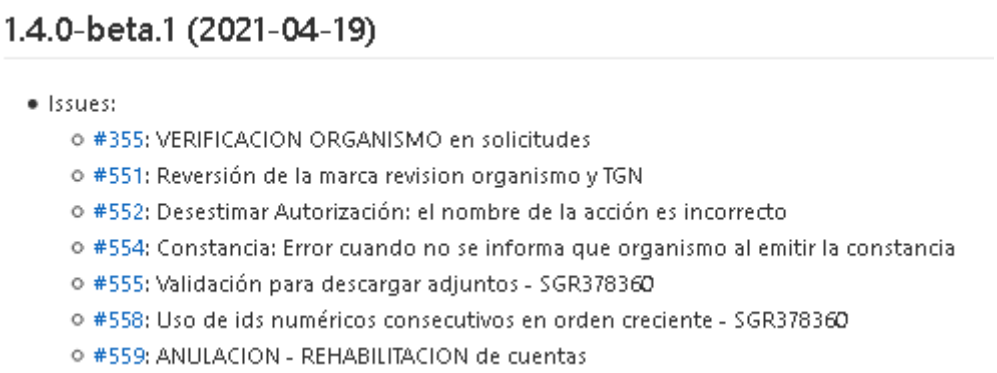

Figura 9 – Detalle de entrega

Donde se ponía el nombre de la versión, fecha en la que se realizó, el contenido y muchas veces era necesario poner dependencias o distintas configuraciones o tareas previas (Figura 10).

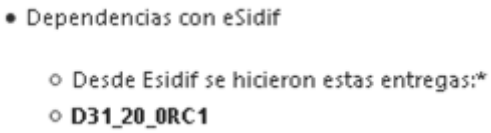

Figura 10 – Ejemplo de dependencias

Otro ejemplo de documentación (Figura 11) de las distintas entregas:

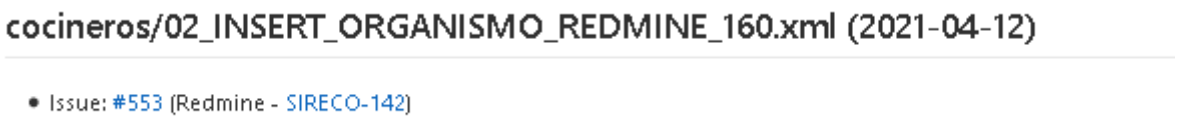

Figura 11 – Ejemplo de documentación

4.1.7 Equipos involucrados y su interacción

Este proyecto estaba constituido por varios grupos de trabajo, los cuales interactuaron uno con el otro para poder llevar a cabo el desarrollo de la aplicación. Entre los equipos más destacados se encontraban:

- Análisis.
- UX/UI (Experiencia de usuario, Interfaz de usuario).
- Desarrollo.
- Testing manual.
- Testing automático.

- CAU (centro de atención a usuarios).
- OKD.

Estos equipos de trabajo son los que interactuaban diariamente para que la aplicación tenga un buen desarrollo y se pueda realizar un producto de buena calidad, y así dejar conforme a los usuarios. A continuación se detalla las tareas realizadas por cada equipo y su interacción con el resto:

Análisis

Al comienzo del proyecto, este grupo fue uno de los encargados de mantener reuniones con el usuario para poder realizar un relevamiento de las necesidades de los usuarios. Fueron observando lo que los usuarios tenían y lo que realmente era lo que necesitaban. Con los requerimientos extraídos de las distintas reuniones, documentaron especificaciones funcionales, poniéndole a cada una de estas una prioridad, la cual ayudó a la hora de realizar las planificaciones y el armado de las etapas de entregas. Hay que mencionar, además era el encargado de priorizar los errores que surgían en producción o en testing

Dependiendo del requerimiento, interactuaba con el equipo de UX/UI, para el armado o modificación del prototipo. En el surgimiento de requerimientos nuevos (previamente analizados y documentados) daba aviso al equipo de desarrollo, luego en reuniones de planificación se realizaba la planificación de todas las tereas (nuevas o errores).

Con el equipo de CAU mantenía una comunicación fluida brindando constante colaboración en conocer las distintas funcionalidades que proveía la aplicación. A su vez Análisis le daba aviso de las funcionalidades nuevas que contienen cada entrega desplegadas en producción.

Si surgía un error en producción la mayoría de las veces, era el encargado de comunicarse con el equipo de desarrollo y exponer lo sucedido donde se buscaba la solución adecuada. Con el equipo de testing mantenía una comunicación puntualmente cuando necesitaban que prueben algún caso en particular o acordar circuitos de pruebas prioritarios en momentos que se los necesitaba.

UX/UI

UX (User Experience) es lo que capta el usuario al interactuar con una aplicación. En cambio, UI (User Interface) es la interfaz con la que se relacionan.

Se utilizó la metodología Design Thinking, encarando el proyecto desde su primera etapa. El equipo se encargó de lograr una comprensión de los requerimientos de los usuarios, recolectando información de distintas reuniones que se realizaron con el área de Análisis, usuarios. Luego de haber investigado la necesidad a través de distintas técnicas de UX, se volcó la información en un prototipo.

Se trabajó colaborativamente para encontrar la mejor solución para el diseño de la interfaz, haciendo foco en la usabilidad[10], en la utilidad, siendo entendible y fácil de usar, respondiendo a las necesidades de los usuarios. La dinámica con ellos fue enriquecedora porque se realizaron ajustes en poco tiempo y a bajo costo, ayudó a la comunicación y a estar alineados entre las distintas áreas. El equipo de UX interactúa con Análisis cuando es necesario encontrar una solución a una parte del diseño. A su vez mantiene una comunicación fluida con el equipo de desarrollo por cualquier cambio que se necesite realizar en la aplicación.

Desarrollo

En un principio, fue el encargado de realizar distintas tomas de decisiones como por ejemplo cómo se iba a mantener la información actualizada que venía desde el e-Sidif. Además era el encargado de plasmar los requerimientos documentados por el grupo de Análisis y llevaba a cabo la implementación del prototipo diseñado por el grupo de UX/UI. A su vez era el responsable de distintas investigaciones sobre posibles tecnologías para la utilización en el desarrollo o soluciones a distintas funcionalidades pedidas por el usuario.

Para cada funcionalidad importante desarrollada se crearon test unitarios (JUnit). A todo esto se agregó la misión de realizar los despliegues y seguimiento de las distintas entregas, para ir instalándolas en el ambiente que corresponda dando aviso a los equipos involucrados.

Periódicamente revisaba el resultado de la ejecución del resultado del testeo hecho por el Sonar, con esto se verificaba que no haya ninguna advertencia sobre el código, si

hubiese alguna, si se podía realizar en ese momento las modificaciones en el código se lo hacía, de lo contrarío se planificaba para tomarlo en algún sprint futuro.

La interacción con los distintos grupos de trabajo era muy fluida ya que estaba en permanente contacto:

- **Análisis:** por si surgía alguna duda sobre algún requerimiento o el seguimiento de alguna tarea.

- **Testing**: mantenía contacto por el surgimiento de dudas sobre la funcionalidad entregada. En el caso de testing automático, algún cambio que pueda afectar el normal funcionamiento de los tests, se le daba aviso de las distintas modificaciones realizadas.

- **UXUX:** su interacción fue más que nada por posibles cambios que se llevaban a cabo referido a la interfaz e interacción del usuario con la aplicación.

- **OKD:** interactuó con este equipo en el caso que se necesitaba realizar el armado, alguna modificación en la configuración del ambiente de OKD o si surgía algún problema con este.

- **CAU**: el vínculo con este equipo era más que nada el soporte que se le daba desde desarrollo ante cualquier duda sobre la funcionalidad, explicando los pasos o posibles alternativas para lograr lo deseado.

Testing manual

Este grupo era el encargado de realizar pruebas manualmente a la aplicación, realizando previamente el diseño de distintos casos de prueba para el testeo de las distintas operatorias entregadas por el equipo de desarrollo. Si bien probaba la parte de la interfaz de usuario, también se centraba en la funcionalidad de cada requerimiento documentado por análisis.

Este interactuaba con el equipo de testing automático donde se ponían de acuerdo para realizar distintas pruebas a la interfaz de usuario. Con los equipos de análisis y desarrollo se comunicaba cuando aparecían dudas sobre alguna funcionalidad o

documentación. Además mantenía una conexión cercana con el grupo de UX/UI por cualquier diferencia que pueda surgir entre la aplicación y el prototipo. Ante cualquier error que detectaban en la funcionalidad o en la interfaz de usuario era el encargado de subir un bug.

Testing automático

Se encargaba de diseñar e implementar los test funcionales (cypress). Estaba en constante comunicación con el equipo de desarrollo no solo por la falla de algún test sino también por el cambio de alguna funcionalidad. Trabajaba en conjunto con el equipo de testing manual definiendo las pruebas que cada uno realizaría sobre la interfaz del usuario.

CAU (Centro de Asistencia al Usuario)

- Tareas:
    - Responsable de capacitar a los usuarios en la utilización de la aplicación.
    - Dieron soporte ante cualquier duda que surgiera por parte del usuario.
    - Realizó los manuales de uso del sistema.
    - Subir peticiones si se detectaba algún error en la aplicación por parte de los usuarios.
    - Informaba a los usuarios ante alguna tarea que puede afectar la disponibilidad del sistema y del contenido de cada entrega nueva instalada.

- La comunicación con los distintos equipos:
    - **Análisis**: la interacción fue constante con este grupo no solo por cualquier duda que pueda surgir con alguna funcionalidad en particular sino que también para estar al tanto por las nuevas funcionalidades que pudiera requerir el usuario. Además de las peticiones que surgían por distintos errores reportados por los usuarios.
    - **Desarrollo:** consultaba a este equipo por cualquier duda con alguna operatoria en el sistema.

OKD

Realizó el armado y las distintas configuraciones en los distintos entornos que requirió el proyecto para ser desplegado con la herramienta OKD, por otro lado dió constante soporte a los equipos ante cualquier duda que surja.

4.1.8 Uso de GitLab

La herramienta GitLab[11], aparte de ser un administrador de repositorios, controlar versiones y código; también ayudó con el seguimiento de las tareas. Para poder tener una mayor organización en GitLab se creó un repositorio “sireco”, el cual tenía dos proyectos, uno para el código, otro para la documentación y temas que no requerían de la ejecución del pipeline:

- **DOCs:** en este proyecto análisis y UX/UI documentaban las distintas tareas que luego el equipo de desarrollo tomaba. Además, se almacenaban documentos de las distintas reuniones tanto internas como las realizadas con el usuario.

- **CODE:** este proyecto se utilizó por el equipo de desarrollo para realizar la codificación y el seguimiento de las distintas tareas.

Se escogió esta división para poder separar el desarrollo de lo que es análisis y UX/UI. Esto viene a que muchas veces se ejecutaban pipelines innecesariamente (por ejemplo si análisis o UX/UI suben un documento, indirectamente es realizar un push sobre el repositorio y dispara un pipeline).

Cada proyecto tenía un tablero para poder llevar una organización de las tareas de cada equipo. El board (tablero) se lo dividió en distintas etapas por la que podía pasar una tarea (issue).

Tablero del proyecto CODE

Sobre este tablero trabajaban distintos equipos (OKD, desarrollo, testing manual y testing automático). Como se muestra en la Figura 12, hay distintas instancias por las cuales podía pasar una tarea. La organización del tablero se hizo con común acuerdo entre los equipos involucrados para poder tener una mejor dinámica, seguimiento y entendimiento de las tareas.

Figura 12 – Board code.

Se realizaba la planificación de las distintas tareas que se desarrollarían durante el sprint siguiente, dejándolas en el “Backlog Desarrollo”. Cada etiqueta tiene un significado y se usan para identificar distintas tareas:

Open	Tareas que se tomarían en los próximos sprints.
OKD	Tareas que se necesitaban llevar a cabo por el equipo de OKD o de desarrollo para la configuración de OKD
Análisis	Tareas que requieran una revisión del equipo de análisis.
Backlog Desarrollo	Tareas que iban a ser tomadas por el equipo de desarrollo durante el sprint.
En desarrollo	Tareas tomadas por el equipo de desarrollo, y se estaban desarrollando en ese momento.

Terminado desarrollo	Tareas terminadas por el equipo de desarrollo, estas se incluían en la siguiente entrega a testing programada.
Disponible Testing	Tareas que estaban disponibles para que testing realice las pruebas.
Backlog Testing	Tareas que el equipo de testing manual iba a probar durante el sprint.
En testing	Tareas que se estaban probando.
TA	Tareas que estaba realizando el equipo de testing automático.
Despliegue Funcional	Tareas necesarias pre despliegue. No siempre se realizaban este tipo de tareas.
Closed	Tareas cerradas, ya sea porque se probaron y estaban ok o porque simplemente se concluyeron.

El board se nutría de distintos tipos de tareas, las cuales tenían diferentes orígenes. Para identificar la procedencia de cada tarea y tener un mejor control de las mismas, se crearon las siguientes etiquetas:

Asociado Análisis	➔ ***Requerimientos de análisis*** ◆ Tareas qué análisis solicitaba realizando una previa documentación de la funcionalidad pedida. Por lo general eran nuevas funcionalidades.
Asociado UXUI	➔ ***UX/UI*** ◆ Tareas generadas por el equipo de UX/UI para ajustar algún cambio en la parte visual de la aplicación.
BUG	➔ ***Bugs*** ◆ Tareas que reflejaban algún error encontrado por el equipo de testing. ◆ Hay que mencionar, además estas eran clasificadas por el equipo de análisis con distintas etiquetas de

	prioridades. (P0, P1, P2, donde P0 es la más importante). Esto ayudó a la organización de las tareas para el día del planning.
Redmine	➔ ***Peticiones*** ◆ Tareas que provenían por pedidos de los usuarios, estas peticiones pasaban por el equipo de análisis para chequear que no haya que ajustar ninguna documentación. ◆ Estas tareas tenían asociada la petición creada en Redmine: Relacionado con SIRECO-135
DYD	➔ ***Internas a desarrollo*** ◆ Tareas que realizaba internamente el equipo de desarrollo (investigación de nuevas tecnologías, soluciones, etc.)

Tablero del proyecto DOCs

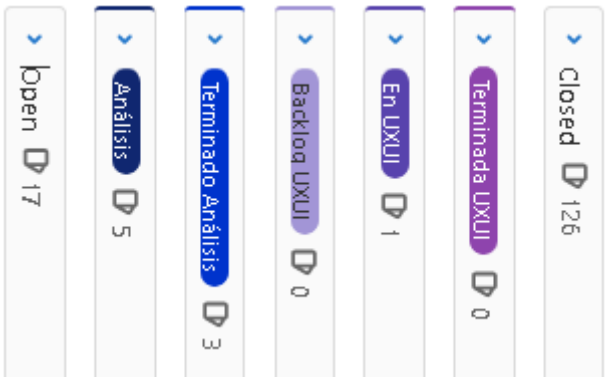

Figura 13 – Board doc.

En este board (tablero) trabajaban los equipos de análisis y de UX/UI. Las tareas que se cargaban no siempre estaban asociadas o requerían la intervención del equipo de desarrollo. El grupo de análisis agregaba distintas tareas (requerimientos nuevos, ajustes de funcionalidad existente, revisión de bugs o peticiones), estas tareas podían quedar en terminadas análisis o bien requerir de la intervención del equipo de UX/UI.

Como se puede observar en la Figura 13, había distintas etiquetas:

Open	Tareas pendientes de tomar.
Análisis	Tareas que fueron tomadas por el equipo de análisis donde se realizó la documentación y/o ajuste si es que lo requería.
Terminado Análisis	Tareas terminadas. (estas podían generar una tarea en el tablero code para que el equipo de desarrollo la tome, mediante planificación de por medio)
Backlog UXUI	Tareas que iban a ser tomadas por el equipo de UX/UI durante el sprint.
En UXUI	Tareas tomadas por el equipo de UX/UI, estaban en proceso.
Terminada UXUI	Tareas terminadas. (Al igual que las tareas terminadas de análisis, estas podían llegar generar una tarea en el tablero code)
Closed	Tareas cerradas, era para las tareas finalizadas o, simplemente se había creado una nueva tarea asociada en el board de code.

Cuando una tarea estaba terminada análisis o UXUI y requería que el equipo de desarrollo participara para la implementación de la funcionalidad documentada en el tablero de documentación. Esta tarea se cerraba y se creaba una tarea nueva en el tablero de desarrollo/testing (board code) asociándola a la tarea cerrada de análisis/UXUI. En el tablero del proyecto de desarrollo/testing quedaba como se muestra en la Figura 14.

Figura 14 – Issue asociado a una petición de usuario y a un issue de análisis

En la descripción quedaba asociada la tarea que le da origen (Figura 15).

Figura 15 – Issue de desarrollo asociado a un issue de análisis

En el tablero de DOCs se veía asociada la tarea de code (Figura 16):

Figura 16 – Asociación del issue de code en el issue de docs

De esta manera, asociando las tareas del tablero de code y el de DOCs, se obtenía un mejor seguimiento de los requerimientos.

4.1.9 Despliegue de la aplicación

La herramienta OKD ayudó a realizar el desarrollo y los despliegues de una forma más ágil y eficiente, gracias a la integración que OKD brinda con jenkins[13]. Donde en el flujo de trabajo se definen ciertos pasos de cómo se realiza el deploy de la aplicación. Se configuró un pipeline, el cual consta de varios pasos y etapas por donde tenía que pasar antes de desplegar en cada ambiente.

Este pipeline no era totalmente automatizado, por decisión de la Organización, en algunos puntos se esperaba la intervención humana. Había diferentes etapas dentro del pipeline, donde cada una tenía pasos para promover la aplicación y llegar a un deploy final:

- **Develop**: Ambiente de desarrollo
  - Build del código
  - Ejecución de test de unidad (desarrollados con JUnit)
  - Deploy de la aplicación en Develop
  - Test funcionales (implementados con cypress)
    - después de realizar los pasos anteriores quedaba a la espera de ser promovido a la siguiente etapa.
- **Testing**: Ambiente de testing, utilizado donde se realizaban pruebas de funcionalidad y de calidad de la aplicación antes de ser usado por el usuario.

- **Staging:** Ambiente de aceptación, es un ambiente utilizado para capacitar al usuario donde puede ir probando distintas funcionalidades, y familiarizándose con la aplicación. Con este entorno el usuario podía aceptar la aplicación antes de su instalación en producción.

- **Producción**: Ambiente productivo, destino final del deploy.

ACLARACIÓN: todos los ambientes tenían configurados los test funcionales pero en Staging y en Producción no se ejecutaban (estaban apagados). Otro tema importante es que al ser tan configurable OKD, daba la posibilidad de apagar los test funcionales tanto en develop como en testing también.

Los pipeline eran disparados por cada push que se realizaba en el repositorio, para poder realizar esta operatoria se utilizaba GitLab, el cual es un controlador de versiones, el cual ayudó al seguimiento de los archivos y de las distintas modificaciones que fue sufriendo la aplicación. GitLab está basado en GIT[12].

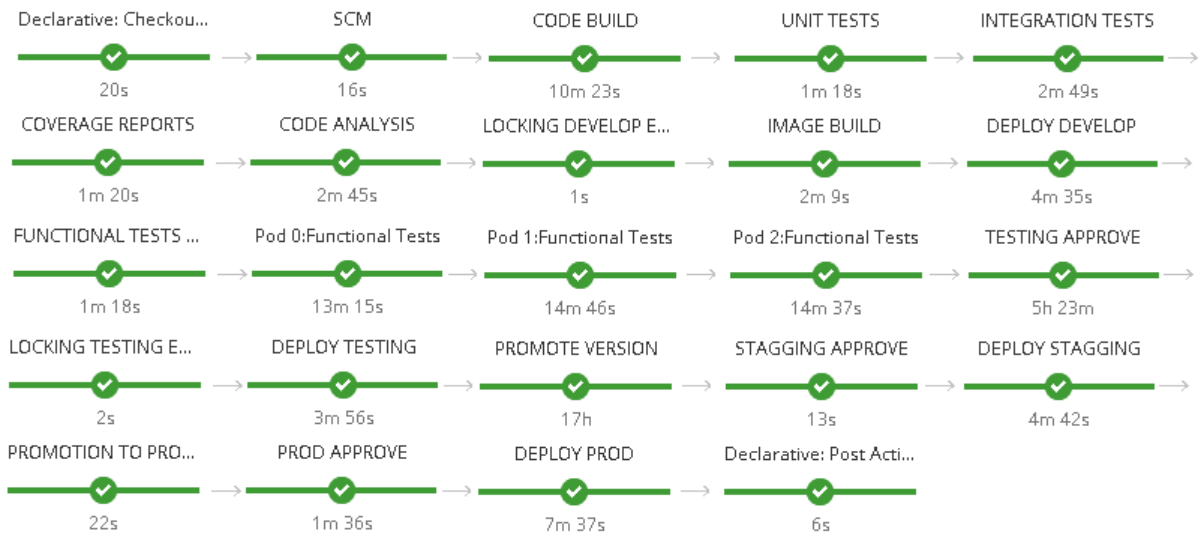

Figura 17 – Ejecución de un Pipeline, pasando por los diferentes entornos (desarrollo, testing y aceptación) llegando a producción como parte final.

Como se comentó más arriba el pipeline (Figura 17) se creaba con cada push, pero el alcance de dónde llega la ejecución dependía de los tags creados.

Posibles nombres de tags:

- **X.Y.Z-beta.W:** todos los tag que contenían “beta” se ejecutaban hasta el ambiente de testing.

- **X.Y.Z-rc.W:** todos los tag que contenían “rc” se ejecutaban hasta el ambiente de staging.
- **X.Y.Z:** los tag con este formato los pipeline se ejecutaban hasta el ambiente de producción.

Por otra parte, el pipeline se podía ejecutar de manera manual desde OKD: “Start Pipeline”

Dónde y cómo se ejecutaba el pipeline

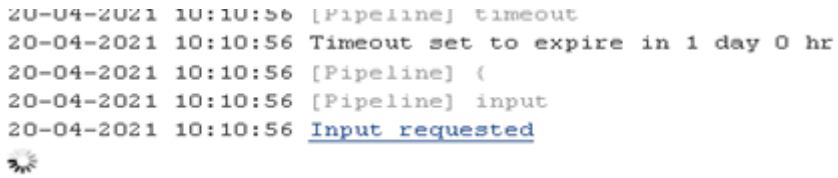

Figura 18 – Ejecución de un pipeline

Se ingresaba un branch o un tag (Figura 19) sobre el cual se quería ejecutar el pipeline y continuaba con la ejecución configurada.

Figura 19 – Ingreso del Branch o Tag sobre donde se ejecuta el Pipeline

4.1.10 Ambiente de Aceptación

Una de las cosas que se incorporó en este proyecto es la posibilidad de darle de forma anticipada al usuario la aplicación para que se fuera familiarizándose. Para esto se creó un ambiente de deploy llamado “Staging”. Este ambiente estaba situado luego del ambiente de testing y antes del de producción.

El equipo de testing probaba distintas entregas que se realizaban de forma incremental para distintas funcionalidades del sistema y daba por aceptadas estas funcionalidades, se generaba una entrega que se desplegaba en el ambiente de Staging pasando previamente por ambiente de desarrollo y testing.

La posibilidad de que los usuarios fueran habituándose con la aplicación, ayudó principalmente a:

- Detectar ciertas funcionalidades que no eran del todo claras para los usuarios.
- Validaciones faltantes a la hora de la creación.
- Reajustes de validaciones que no eran correctas
- Detectar datos incorrectos de los que se iban a migrar

Todo lo detectado por los usuarios en esas pruebas se fue planificando y ajustando en entregas posteriores.

Capítulo 5

5.1 Descripción del sistema

En primer lugar, cabe mencionar que las cuentas oficiales son cuentas de los organismos del Sector Público Nacional incluidos en el artículo N°8 de la Ley N°24.156, a través de las cuales se gestionan recursos y gastos que están dentro del presupuesto de la Administración Nacional. Estas cuentas pueden ser bancarias o nominales.

Las cuentas bancarias son informadas por los organismos mediante el alta de cuentas y las nominales son cuentas creadas por la Tesorería (además, puede dar de alta cuentas bancarias), estas cuentas no son reales (no pertenecen a un banco), sino que se encuentran únicamente en la Tesorería y se utilizan como recaudadoras (ingresos de recursos) pero no generan movimientos de fondos [14].

Como anteriormente se mencionó esta aplicación permite llevar el registro de cuentas oficiales donde la Tesorería puede realizar consultas de todas las cuentas en cualquier momento. El Organismo puede consultar las cuentas que son de su titularidad, salvo que la Tesorería las haya dado de alta durante el período de empadronamiento vigente y este siga habilitado. Con la finalización del período, el titular puede consultar todas sus cuentas. Algunos de los datos de estas son solo visibles para la Tesorería.

Figura 20 – Resultado de la consulta cuentas

El resultado de la consulta (Figura 20) puede exportarse a PDF y/o XLSX. Esta operación sobre los datos da la posibilidad de seleccionar previamente las columnas que se quieren visualizar en los distintos archivos, dando la posibilidad de armar reportes dinámicos y obtener la información de diferentes maneras.

A su vez, el sistema permite que se ingresen solicitudes de autorización de aperturas de cuentas en una Entidad Bancaria. Estos pedidos los realizan los distintos

Organismos y solo lo pueden cargar en el sistema la Tesorería. Para estos casos solo se completan algunos datos (Banco, Sucursal, Denominación, Organismo, CUIT, Tipo de cuenta, Moneda, Fuente de financiamiento, Clase, Grupo Fuco, Fecha de autorización, nro. de archivo, observación, etc.). El número de cuenta no se informa, esto se debe a que no se tiene este dato hasta que no se abra en el banco.

Con respecto a las modificaciones de las cuentas, las pueden realizar tanto los Organismos como la Tesorería. Donde los titulares sólo pueden modificar y revisar sus cuentas, siempre y cuando se encuentre dentro del período de empadronamiento habilitado. Como en la creación, los titulares sólo pueden modificar algunos datos. Hay cuentas que no pueden ser editadas por los organismos, como por ejemplo: con estado anulado, revisada por el organismo, cuentas nominales, solicitudes de autorización de apertura de cuenta desestimadas.

Las autorizaciones de aperturas de cuentas, solo pueden ser editadas por la Tesorería, si el titular quiere editarlas está obligado a cargar el número de cuenta (también lo puede realizar la Tesorería en cualquier momento del año). Muchas veces la cuenta no se abre en el banco, por lo que es necesario registrarlo en el sistema. Esto se hace mediante la “Desestimación de autorización” (es una acción habilitada para el Organismo, dejando seteada la fecha de baja con la fecha del día y la marca “Revisada Tesorería” en “NO”. La fecha de baja también la puede cargar la Tesorería.

Cada cuenta posee unas marcas de revisión, “Revisada por Tesorería” y “Revisada por Organismo”, cuando es creada por un organismo estas marcas quedan en “NO”. Para el caso en que la Tesorería dé de alta, estas marcas quedan ambas en “SI”. Ante cualquier cambio que sufra cada cuenta por parte del titular, se chequea que lo modificado no sean datos sensibles para la Tesorería, si esto es así, la marca de “Revisada por Tesorería” se pone en “NO”. Para las modificaciones que realiza la Tesorería, las marcas se mantienen en “SI”. Más adelante, cuando el Organismo obtiene el número de cuenta en la entidad bancaria, se carga este dato, la fecha de apertura y otros datos significativos.

Uno de los objetivos de este sistema es permitir a los organismos revisar las cuentas en un momento determinado donde se comprueban que estas estén en uso (de lo contrario de da de baja) y/o los datos estén correctos. El momento que se le da al usuario para poder modificar las mismas y luego revisarlas se denomina Periodo de Empadronamiento y este tiempo es dispuesto por la Tesorería. Una vez finalizado este

período, los organismos solo pueden realizar consultas sobre sus cuentas, constancias y entidades básicas.

Para la revisión de las cuentas por parte del titular, primero comprueba que los datos de la cuenta estén correctos, para luego marcar como revisada la cuenta. Esta revisión se realiza mediante una acción, lo que deja la marca “Revisada Organismo” en “SI”. Se puede realizar siempre y cuando el período de empadronamiento vigente esté habilitado y el organismo no haya generado la “Constancia de empadronamiento” para dicho período.

Las cuentas que están revisadas dejan de estar editables para los organismos y se visualiza la marca de revisada (Figura 21).

Figura 21 – Botón de revisar cuenta inhabilitado, y check en de revisada en SI

Por otro lado, existe una acción de “Revertir revisión de cuenta”, donde esta funcionalidad permite al titular sacar la marca de revisada y poder realizar modificaciones faltantes a la cuenta. Se puede revertir una revisión siempre y cuando el período de empadronamiento esté vigente y el Organismo no tenga constancia generada para ese período.

Así como el Organismo verifica sus cuentas, también la Tesorería realiza la misma operación para cada cuenta que tiene la marca “Revisada Tesorería” en “NO”. Se puede llevar a cabo la revisión, siempre y cuando el titular de la cuenta haya revisado su cuenta (revisada en “SI”). La Tesorería puede ejecutar la acción en cualquier momento, haya o no un período de empadronamiento habilitado. Además, puede revertir esa marca en cualquier momento sin ninguna limitación.

Cada Organismo, después de revisar su stock de cuentas, debe emitir una constancia de empadronamiento, la cual deja testimonio que se finalizó con la revisión de las cuentas. Esta constancia se debe generar dentro del período de empadronamiento vigente, el formato es PDF. Existe sólo una constancia para cada Organismo dentro de un periodo de empadronamiento.

Tipo	Organismo	Periodo de Empadronamiento	Fecha de emisión	Estado	Usuario	Adjunto
AC	330	09/03/2021 - 09/07/2021	28/05/2021	Generada	DNI33033000	

Figura 22 – Grilla con la constancia “Generada” para el Organismo 330

Para la generación de esta constancia no se tienen en consideración las cuentas anuladas, no vigentes, desestimadas, y aquellas que fueron dadas de alta por la Tesorería durante el periodo vigente de empadronamiento. Y la puede realizar el Organismo pero no puede anularla (sólo la Tesorería puede realizar esa anulación).

Si la Tesorería anula una constancia, y se realiza dentro del período de empadronamiento vigente habilitado, el Organismo dispone, nuevamente, de la posibilidad de edición de cuentas y/o volver a generar la constancia. Una constancia anulada no se puede rehabilitar, se debe emitir una vez más.

5.1.1 Entidades Básicas

Una entidad básica o también conocida como tabla básica, proporciona datos necesarios para las cuentas oficiales. Estos datos se utilizan para completar y/o validar la información ingresada de las cuentas. Estas entidades pueden ser consultadas en cualquier momento tanto por la Tesorería como por los organismos. Además de permitir la explotación a Excel y PDF de los datos de cada entidad.

Hay entidades cuyos datos residen en SIRECO y otras se consultan desde e-Sidif mediante una vinculación por API Rest.

Figura 23 – Visualización de las Entidades Básicas

Entidades específicas de SIRECO

- **Tipo Organismo:** Determina los diferentes tipos de organismos de las cuentas:
  - Administración Central (AC)
  - Organismos Descentralizados (OD)
  - Instituciones de Seguridad Social (ISS)
  - Empresas Públicas (EP)
  - Fondos Fiduciarios (FF)
  - Otros Entes (OE)
  - Universidades (UUNN)
  - Sector Público Financiero no bancario (SPF)
  - Sin clasificar (OTR)

- **Organismo:** Identifica los titulares de las cuentas. Los organismos pueden ser Servicios de Administración Financiera (SAF, cuyos datos fueron extraídos de e-Sidif) y Empresas, Fondos Fiduciarios, Universidades, Otros Entes (obtenidos desde el Sistema de Información Financiera para Empresas Públicas, Fondos Fiduciarios, entes excluidos del Presupuesto de la Administración Nacional y Universidades Nacionales[15]).

- **Artículo 8:** Clasificación de los organismos conforme a los incisos definidos en el artículo n° 8 de la Ley de Administración Financiera n° 24.156. Con este dato se valida si las cuentas requieren de la autorización previa de la Tesorería y cuáles sólo necesitan ser informadas.

- **Tipo Cuenta:** Determina el tipo de cuenta:
  - Cuenta Corriente
  - Cuenta Corriente Especial
  - Cuenta Nominal
  - Cuenta Comitente
  - Cuenta de Garantía
  - Cuenta Especial

  Los organismos solo pueden llevar a cabo operaciones con los tipos de cuentas: Comitente, Corriente y Especial.

- **Clase:** Identifica los tipos de gestiones que se realizan mediante el uso de la cuenta:
  - Recaudadora Pura
  - Recaudadora Secundaria
  - Cuenta Pagadora Principal
  - Cuenta Nominal
  - Haberes
  - Fondo Fiduciario
  - Cuenta Custodia
  - Préstamos / Progr./Donac.
  - Unidades Desconcentradas

- **Tipo de Identificador Bancario:** Especifica el tipo identificador bancario. Utilizando IBAN y CLABE para bancos extranjeros y CBU para bancos argentinos.

- **Grupo Fuco:** Determina el tipo de cuenta en consonancia con el Fondo Unificado de Cuentas Oficiales. Este atributo se utiliza únicamente en cuentas corrientes del BNA (Banco de la Nación Argentina).

Entidades consultadas desde e-Sidif

- **Banco:** Determina a los bancos locales o extranjeros donde las cuentas fueron abiertas.

- **Sucursal:** Específica a la sucursal del banco de una cuenta oficial.

- **País:** Indica el país donde está ubicada la sucursal del banco de una cuenta.

- **Provincia:** Identifica la provincia donde se encuentra radicada la sucursal de un banco de Argentina.

- **Fuente de Financiamiento:** Es un clasificador presupuestario y el propósito es diferenciar el origen de los recursos que forman parte de los saldos de las cuentas.

- **Moneda:** Indica la moneda de la cuenta oficial de acuerdo con la Norma ISO 4217[16].

5.1.2 Cuenta

A lo mencionado anteriormente de la operatoria sobre las cuentas, se debe agregar que el stock inicial de cuentas oficiales del sistema se migró desde el sistema anterior. Se utilizó el ambiente de staging para realizar una comprobación y corrección de datos (si es que se necesitaba) por parte de los usuarios de la tesorería. Con esto se logró pulir el conjunto de datos de las cuentas.

Los datos de las cuentas oficiales se dividen en tres grupos: generales, firmantes y archivos adjuntos.

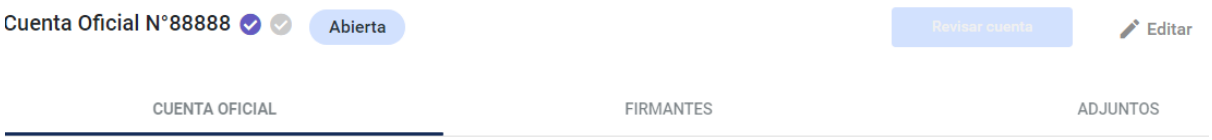

Figura 24 – Visualización en tres solapas de los datos de una cuenta oficial

Donde los generales (Cuenta Oficial), se completan y/o visualizan datos del Organismo, responsable operativo, identificación de la cuenta, fechas, otros datos, información relacionada con la tesorería y datos de auditoría.

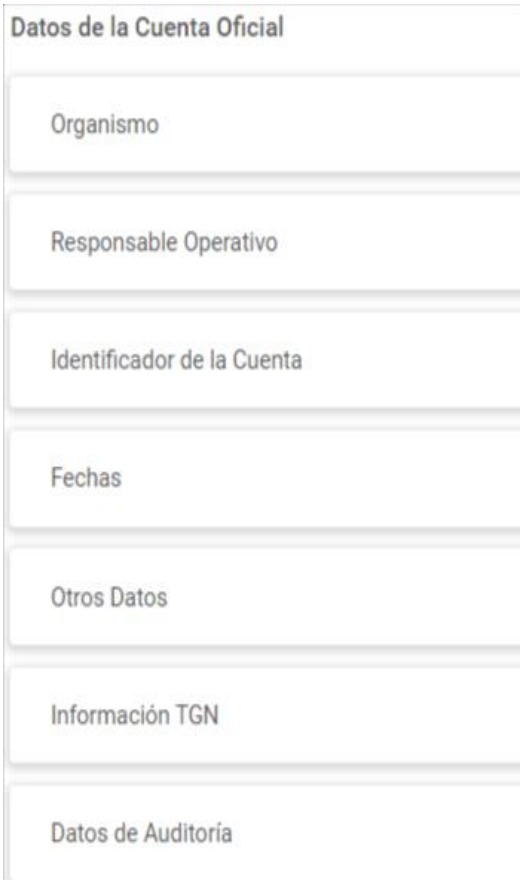

Figura 25 – Datos generales de una cuenta oficial

Un firmante es la persona que se especifica en el banco como el responsable de la cuenta, además si tienen cheques es el encargado de firmarlos.

Figura 26 – Datos de los firmantes de una cuenta oficial

Los adjuntos son archivos que incorporan el Organismo y/o la Tesorería con distinta información relacionada con la cuenta.

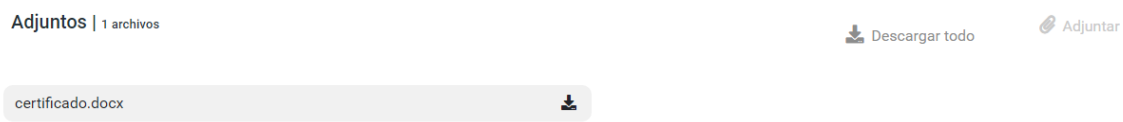

Figura 27 – Archivos adjuntos de una cuenta oficial

La unicidad de una cuenta oficial, por lo general, es banco, sucursal y número de cuenta. En cambio, hay circunstancias en donde la cuenta puede no tener número informado o bien puede darse la posibilidad de ingresar cuentas duplicadas (mismo banco, sucursal y número).

La duplicidad a nivel banco, sucursal y número sucede con los bancos de origen extranjeros y para cuentas nacionales de los bancos Patagonia y del Central. Para el caso anterior el sistema advierte de la existencia de duplicidad y el usuario decide si quiere o no guardar esa cuenta. Sin embargo, para cuentas que no están dentro de las condiciones anteriores mencionadas, la validación es restrictiva.

Los datos como banco, sucursal, fuente financiamiento y moneda sólo se almacena el código de cada uno en la cuenta, si es necesario algún otro dato de estas entidades, por ejemplo la descripción, se consultan en el e-Sidif. Con esto se podía tener un problema de performance en la consulta de cuentas, ya que cómo se ve en la Figura 28 se muestra el código y la descripción del banco y de la sucursal en cada cuenta.

Figura 28 – Grilla de resultados de búsquedas de cuentas oficiales

Para lograr un buen rendimiento en la búsqueda y visualización de la grilla de las cuentas oficiales se realizaron dos tablas intermedias, donde se almacenan los códigos y las descripciones de los bancos y sucursales utilizadas por las cuentas. Estas tablas se completan a partir de la creación y/o modificación de las mismas.

El proceso se realiza a partir de la creación y/o actualización de una cuenta donde:

1- Se consulta en e-Sidif la descripción para el banco y la sucursal.

2- Se verifica la existencia del banco y la sucursal en las tablas intermedias.

3 - Si los datos se encuentran en las tablas, sólo se actualiza la descripción, de lo contrario se guarda el código y descripción.

Las cuentas pueden ser anuladas sólo por la Tesorería, esto ocurre cuando la cuenta tiene algún dato incorrecto y no va a ser utilizada. Además, si puede volver atrás esa anulación, que se realiza con la rehabilitación.

Estado de una cuenta

Cada cuenta posee un estado, el cual está sujeto a distintos atributos fechas o la combinación entre estas. Es una propiedad no es editable, sino que es calculada ante cualquier cambio de dato de la cuenta y/o diariamente se ejecuta un proceso el cual recalcula el estados para todas las cuentas. Este dato se persiste en la base para tener mejor performance y no tener que ejecutar un algoritmo ante cualquier consulta o modificación de una cuenta. No hay cuentas sin estado.

El proceso mencionado anteriormente, se ejecuta diariamente comparando algunas fechas de la cuenta, sumándole la fecha del día para algunos casos, de esta comparación pueden surgir los siguientes estados para cada cuenta:

- **Autorizada:** Toma este estado cuando no pasó un mes de la autorización de apertura de una solicitud de una cuenta oficial por parte de la Tesorería (Fecha de autorización informada + 30 días es menor o igual a la fecha de hoy).

- **Autorizada vencida:** Adquiere este estado cuando pasaron 30 días desde la autorización de apertura y el organismo no informó el número de cuenta asignado (Fecha de autorización informada + 30 días es mayor a la fecha de hoy).

- **Autorizada no abierta:** Alcanza este estado cuando tiene la fecha de baja informada. Esto quiere decir que el organismo no llevó, ni llevará a cabo la apertura de la cuenta (Fecha de baja informada).

Los estados anteriores nombrados son estados en los cuales aún no están informados los datos sensibles como el tipo identificador (CBU, CLABE e IBAN), número de identificador, número de cuenta, entre otros. A continuación se detallan los estados que parten de cuentas con los datos completos.

- **Abierta:** Adquiere este estado cuando la cuenta tiene fecha de apertura, número de cuenta y si tiene fecha de vencimiento, la misma es mayor o igual a la fecha del día.

- **Cuenta Vencida:** Alcanza este estado cuando una cuenta tiene fecha de vencimiento informada, es menor a la actual y no se completaron las fechas de anulación, cierre y de baja.

- **Informada:** La cuenta obtiene este estado cuando tenga informada la fecha de notificación y las fechas de apertura, no convalidación y convalidación en nulo.

- **Convalidada:** Tiene este estado cuando la cuenta no está vencida y la fecha de convalidación está informada, además, las fecha de anulación y cierre no tiene valor.

- **No Convalidada:** Similar al estado anterior, pero con la diferencia que la fecha informada debe ser la de No convalidación.

- **Cerrada:** Para este estado se tiene que informar la fecha de cierre pero la de anulación no debe tener valor.

- **Anulada:** La cuenta se encuentra en este estado cuando la fecha de anulación tiene valor.

Historial

La Tesorería, además, de poder realizar altas, bajas y modificaciones en cualquier momento sin tener en cuenta la existencia de un periodo de empadronamiento habilitado, puede visualizar las modificaciones que fueron atravesando las cuentas mediante un historial.

El sistema permite tener un historial con el registro de las distintas modificaciones que sufre cada cuenta. Por cada modificación que se realiza sobre la cuenta se guarda una foto de los datos. Esta historia es visible sólo para la Tesorería, y puede consultarse desde una cuenta en particular o desde el menú principal. Este se puede exportar a Excel.

Historial | 4 elementos

Filtro Exportar

Organismo	CUIT Org.	Tipo Org.	Art.8 Ley 24.156	Org. CUT	CUIT Resp. Op.	Cuenta Oficial	Denomin
330 - Ministerio de Educación	30628540787	AC	a	Si		88888	hhhhhhhl
330 - Ministerio de Educación	30628540787	AC	a	Si		88888	hhhhhhhl
330 - Ministerio de Educación	30628540787	AC	a	Si		88888	hhhhhhhl
330 - Ministerio de Educación	30628540787	AC	a	Si		88888	hhhhhhhl

Figura 29 – Grilla de historial de una cuenta

5.1.3 Migración de datos para la inicialización del contexto

Como se fue viendo durante todo este trabajo se utilizan distintos datos, entre los cuales hubo que realizar migraciones de otros sistemas. Como por ejemplo los datos de los Organismos (que se obtuvieron desde e-Sidif o de SIFEP). Pero hubo más migraciones de los distintos datos que son utilizados para la consulta y gestión de las cuentas oficiales, entre estos se encuentran las tablas propias de SIRECO. Estos datos fueron depurados entre el equipo de análisis y los usuarios de la Tesorería, partiendo de los datos contenidos en el sistema anterior. Una vez hecha esta limpieza de las entidades básicas, se realizaron distintos script donde se fue armando el momento cero de la aplicación.

Luego de la inicialización de las entidades básicas, se trabajó en la migración de las cuentas desde el sistema anterior. Esto llevó consigo, la depuración y revisión manual de cada cuenta por parte de la Tesorería. Fue un arduo trabajo, donde se obtuvo una planilla Excel, la cual se utilizó para realizar la importación de las mismas en el sistema nuevo. La funcionalidad de importación de cuentas se utilizó por única vez en la aplicación, donde se insertaron todas las cuentas que la Tesorería controló para obtener el momento cero antes del startup del sistema.

Para el momento cero se plantearon dos posibilidades: realizar la migración desde el sistema anterior o desde otros, o insertar por aplicación todos los datos desde cero de forma manual. Para esto se acordó con la Tesorería realizar las distintas depuraciones y migraciones.

5.1.4 Diagrama

En el siguiente diagrama se puede observar la Cuenta con sus distintos atributos, entre los que se encuentran las entidades básicas, también se diferencian las entidades específicas de SIRECO y las que son consultadas desde e-Sidif.

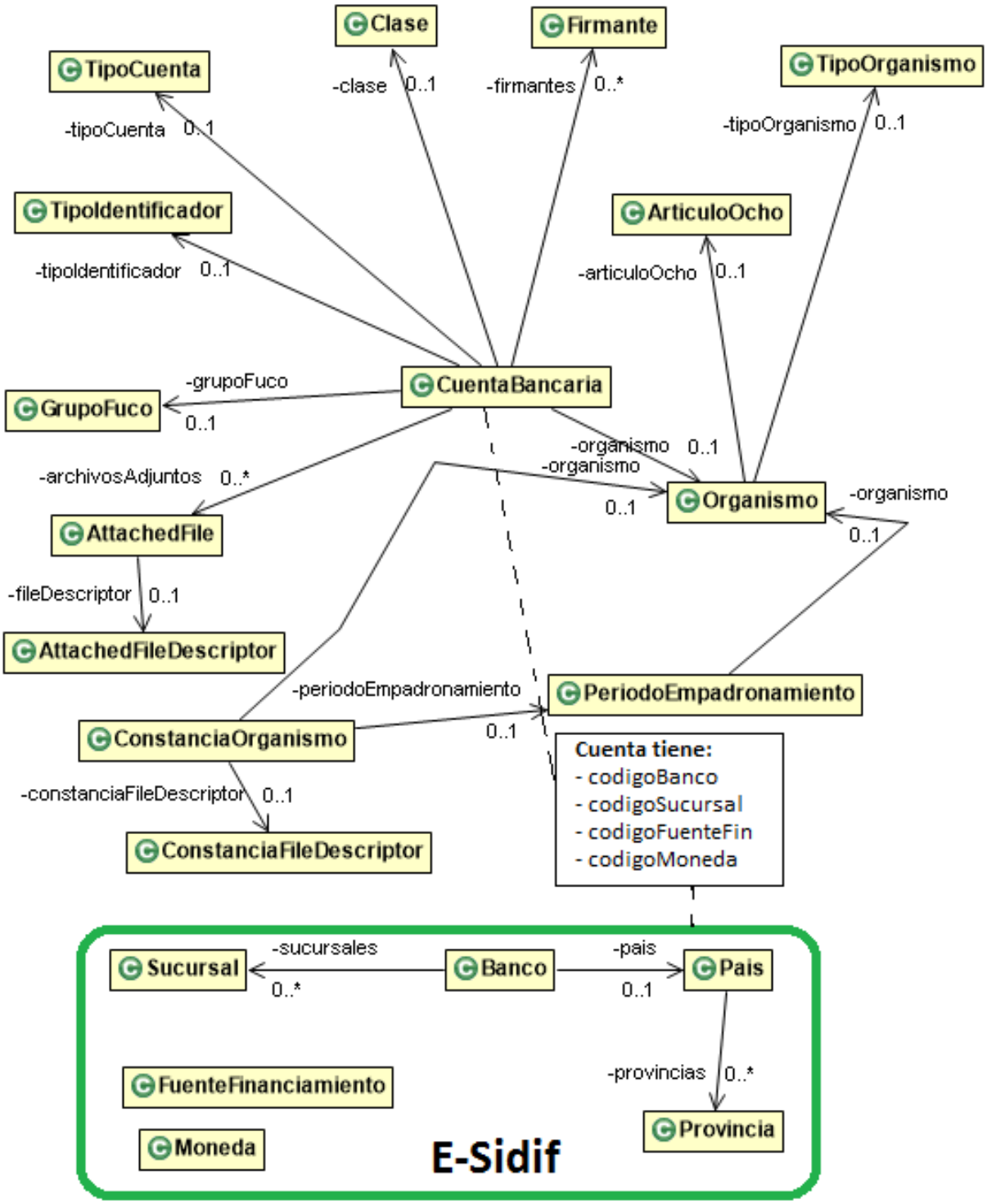

Figura 30 – Diagrama de las distintas entidades de SIRECO y e-Sidif

Capítulo 6

6.1 Conclusión

En este trabajo se comenzó con una introducción sobre los inicios de la DGSIAF, el surgimiento de e-Sidif y los distintos sistemas “satélites”, en especial se apuntó al Sistema de Registro de Cuentas Oficiales del Sector Público Nacional (SIRECO); para luego pasar a mostrar distintas metrologías que se utilizaron durante el proceso de prototipado, análisis, desarrollo y puesta en marcha del sistema.

Si bien dentro de la Organización se comenzaban a utilizaban nuevas tecnologías y herramientas, SIRECO fue el punta pie inicial para poder iniciar un cambio en los nuevos desarrollos, ya que se aplicaron nuevas tecnologías para la implementación de la aplicación sino que también se cambió la forma de despliegues de aplicaciones, utilizando nuevas tecnologías y herramientas.

Por un lado la implementación del sistema SIRECO trajo a los usuarios una renovación tecnológica y de procesos realizados en el registro de cuentas, por otro lado, con este aplicativo se le alivió el trabajo a la Tesorería y se involucró a los Organismos dejándoles realizar distintas operaciones sobre sus cuentas.

SIRECO dejó un precedente positivo con respecto a los procesos utilizados, el cual se puede tomar la experiencia obtenida y los procesos de trabajos utilizados y emplearlos en las aplicaciones que van a surgir en un futuro. La incorporación de grupo UXUI, el hecho de hacer partícipe al Usuario dentro de todo el desarrollo del aplicativo, utilización de nuevas herramientas y tecnologías tanto para la implementación, documentación, nueva forma de despliegues, comunicación (Mattermost, redmine), entre otros.

6.2 Trabajos futuros

Para lograr optimizar la gestión del usuario, se podría plantear algunas mejoras o implementaciones nuevas. A su vez, se pueden sumar mejoras en el proceso de la construcción de un sistema dentro de la DGSIAF. A continuación se enumeran algunas:

- **Gestión de las entidades básicas:** Alta, baja y modificación de las distintas entidades básicas propias de SIRECO.

- **Gestión de periodo de empadronamiento:** Alta, baja y modificación de los periodos de empadronamiento.

- **Circuito para la gestión de solicitud de alta de nuevas cuentas bancarias:** Un Organismo pueda solicitar la apertura mediante la aplicación.

- **Actualización de datos de entidades básicas de SIRECO provenientes de e-Sidif y de SIFEP:** Como por ejemplo Organismos.

- **Reportes de Cuentas Oficiales**.

- **Interoperabilidad con Cuentas Operativas de e-Sidif:** A la hora de gestionar cuentas en SIRECO, mediante la comunicación entre ambos sistemas reflejar esta gestión en e-Sidif.

- **Validación de la cuenta bancaria con Interbanking** [17]**:** En la gestión de cuentas, validar el CBU y CUIT del titular mediante la comunicación por WS con Interbanking.

- **Métricas:** Con estas se logra una visión del funcionamiento de las distintas operaciones. Dentro de la organización se utiliza Prometheus [18].

- **Biblioteca de componentes DGSIAF:** Implementar microfrontends reutilizables.

- **SQL “cocineros” dinámicos:** Si bien no es algo frecuente, pero existen casos aislados donde se necesita modificar la base de datos por distintos motivos, donde sería necesario poder contar con un SQL dinámico.

Bibliografía

[1] - *Decreto 1344/2007: http://servicios.infoleg.gob.ar/infolegInternet/anexos/130000-134999/133006/norma.htm*

[2] - *Ley 24.156: http://servicios.infoleg.gob.ar/infolegInternet/anexos/0-4999/554/texact.htm#1*

[3] - *Disposición N° 5 2010/TGN: http://forotgn.mecon.gov.ar/normativa/disposiciones/disposiciones_tgn/2010/disp20105.pdf*

[4] - *DGSIAF: https://dgsiaf.mecon.gov.ar/ , se migró en el año 2021 a https://www.argentina.gob.ar/economia/sechacienda/dgsiaf*

[5] - *Manifiesto ágil, Principios: http://agilemanifesto.org/iso/es/principles.html*

[6] - *Scrum: https://scrumguides.org/download.html*

[7] - *DevOps: https://about.gitlab.com/topics/devops/*

[8] - *DevOps: https://www.redhat.com/*

[9] - *Design Thinking: https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process*

[10] - *Krug, Steve . No me hagas pensar. Una aproximación a la usabilidad en la web. 2da Edición. ISBN 84-8322-286-8*

[11] - *GitLab: https://about.gitlab.com/*

[12] - *Git: https://git-scm.com/*

[13] - *Jenkins: https://www.jenkins.io/*

[14] - *Disposiciones CGN N° 13 y TGN N° 5 del 16/02/96: https://www.economia.gob.ar/hacienda/cgn/normas/disposiciones/1996/disp13/dis13.htm*

[15] - *SIFEP: https://www.argentina.gob.ar/economia/sechacienda/dgsiaf/sifep*

[16] - *Norma ISO 4217: https://www.iso.org/iso-4217-currency-codes.html*

[17] - *Interbanking: ttps://www.interbanking.com.ar/*

[18] - *Prometheus: https://prometheus.io/*

Anexo de tecnología y herramientas utilizadas

El aplicativo cuenta principalmente con un frontend donde se utilizó angular, angular material, bootstrap, entre otras tecnologías. Está compuesto por distintas pantallas, las cuales se van a detallar más adelante. El frontend se comunica con el backend utilizando API Rest.

El backend realiza todas las interacciones con la base de datos Oracle 12 y a su vez se vincula con el sistema e-Sidif mediante distintos API Rest, para poder obtener información, la cual es útil para la gestión de las cuentas oficiales. Además, para la autenticación y autorización del usuario, el backend se comunica con un servicio web de AFIP y de e-Sidif.

En esta sección se nombraran algunas de las tecnologías y herramientas más destacadas en el proyecto, las cuales fueron utilizadas durante el desarrollo de esta aplicación.

HTML5

HTML significa Hipertext Markup Language, se utiliza para estructurar las páginas web utilizando etiquetas o tags, la estructura es de tipo árbol. Es un lenguaje de marcas, el cual es interpretado por los browsers.

CSS

Además del HTML, el cual ayuda a estructurar una página web, también se tiene que poder decorarla, para esto se necesita darle un estilo. El CSS (Hoja de Estilo en Cascada) es un lenguaje que ayuda a darle un mejor aspecto a un documento estructurado, dándole mejor legibilidad. Con los estilos se definen tipos de letras, colores, alineación de partes dentro de la página, dejando una mejor apariencia del documento.

Bootstrap

Framework que ayuda al armado del diseño del frontend de una aplicación, es más fácil y simple de utilizar, ya que proporciona una gran variedad de elementos, entre los

cuales se encuentran plantillas de diseños, botones, formularios, tablas, y muchos otros más, también adiciona varios componentes de javascript. Facilita el desarrollo responsivo, haciendo que las aplicaciones se puedan ajustar al dispositivo en el cual se mire (computadoras, teléfonos, tablets).
URL: https://getbootstrap.com/

Angular 10

Es un framework de código abierto empleado para el desarrollo de páginas web, utiliza el patrón MVC (Model View Controller). Las aplicaciones desarrolladas con este framework son de tipo SPA (Single Page Application). Y permite programar componentes y servicios reutilizables. Angular está sentado en TypeScript.
URL: https://angular.io/

Angular Material

Es una librería de estilos, similar a Bootstrap, la cual se apoya en Material Design. Esta librería está desarrollada por el grupo de Angular. Provee distintos componentes, los cuales son bloques interactivos que ayudan en la creación de la interfaz de usuario.
URLs: https://material.angular.io/ y https://material.io/design

Node.js

Es un entorno de ejecución multiplataforma para la capa del servidor, el cual Angular lo utiliza para correr, se basa en el lenguaje de programación JavaScript y es de código abierto. Proporciona un conjunto de primitivas de E/S asincrónicas en una arquitectura orientada a eventos.
URL:https://nodejs.org/

NPM

NPM (Node Package Manager) es el manejador de paquetes y librerías de Node.js, siendo un repositorio de código abierto para proyectos Node.js, en el cual se pueden compartir distintos recursos, además se puede interactuar con el repositorio mediante las

líneas de comandos para poder realizar distintas operaciones con los paquetes, versiones y/o dependencias.
URL: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

Java

Para el backend de la aplicación se desarrolló en Java con la utilización del framework Spring Boot. Para la compilación del código y manejo de dependencias se utiliza la herramienta Gradle, además utiliza JDK 11 (Java Development Kit - librerías y programas que ayudan al desarrollo en el lenguaje Java).
URL: https://spring.io/projects/spring-boot y https://gradle.org/

Liquibase

Herramienta utilizada para el seguimiento, versionado e implementación de cambios en la base de datos. Además permite probar con facilidad las modificaciones que se realizan en la base de datos, ya que permite crear puntos de retornos, volverlos atrás con gran facilidad.
URL: https://www.liquibase.org/

Oracle

Oracle es un sistema de gestión de bases de datos de tipo objeto relacional. La base de datos es tradicional ya que sus datos están almacenados en tablas y estas contienen filas por cada dato. Para esta aplicación se utiliza Oracle 12.

OKD

"OKD es una distribución de Kubernetes optimizada para el desarrollo continuo de aplicaciones y la implementación de múltiples despliegues. OKD agrega herramientas centradas en las operaciones y el desarrollador sobre Kubernetes para permitir el desarrollo rápido de aplicaciones, la implementación y el escalado fácil..." Red Hat OpenShift es la impulsora de esta distribución.
URL: https://www.okd.io/

Herramientas para el testeo

Cypress es una herramienta que se utiliza para el testeo del frontend de la aplicación. Las pruebas realizadas se ejecutan automáticamente con cada pipeline en OKD. Además de testear el frontend con cypress, se utiliza JUnit y Sonar, donde JUnit se utiliza para el testeo funcional de la aplicación y Sonar para el testeo estático del código.
URLs: https://www.cypress.io/, https://junit.org/junit5/ y https://www.sonarqube.org/

Servicio REST

Servicio REST (Transferencia de Estado Representacional), se utiliza para la comunicación entre dos sistemas que utilicen HTTP. En el aplicativo, esta arquitectura se utilizó para la comunicación entre el frontend y el backend. Además se utilizó para la vinculación con el sistema e-Sidif, para realizar distintas consultas necesarias para la gestión de las cuentas.

Redmine

"Redmine es una aplicación web de gestión de proyectos flexibles. Escrito con el marco de Ruby on Rails, es multiplataforma." Es utilizada
URL: https://www.redmine.org/

Mattermost

Herramienta utilizada para la comunicación entre los integrantes del equipo, es un chat el cual permite estar más conectados, tener búsquedas de información dentro de las distintas conversaciones, enviar archivos; evitando el desorden de lo que puede llevar el uso excesivo de mails. Esto permitió un aumento de productividad en las distintas etapas del proyecto.
URL: https://mattermost.com/

GitLab

Es una plataforma DevOps que ayuda al desarrollo ágil, integración continua, controlar las distintas versiones, seguimiento de los archivos y de las distintas

modificaciones que fueron sufriendo, entre otras. Es un administrador de repositorios, y está basado en GIT.
URL: https://about.gitlab.com/ y https://git-scm.com/